GUARANTEED ROBOTIC PLANNING AND EXPLORATION IN UNKNOWN ENVIRONMENTS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Alexander Ivanov Ivanov

May 2019

GUARANTEED ROBOTIC PLANNING AND EXPLORATION IN UNKNOWN ENVIRONMENTS

Alexander Ivanov Ivanov, Ph.D.

Cornell University 2019

Robots have become ubiquitous in modern life, but their autonomous application has, thus far, been relegated to well known, well structured, or benign environments. Practical guarantees for robotic path planning in unknown environments have been elusive. This work explores three distinct problems in robotic planning and exploration in unknown environments and their interconnections.

First, information gathering is considered in an environment with known obstacles. An algorithm is developed which solves the information gathering problem while providing a probabilistic guarantee on localization. The algorithm's behavior is analyzed, and a practical demonstration is presented.

Second, multi-robot exploration and tracking of multiple objects is explored. A centralized planning algorithm is developed which provides a guarantee on maintaining tracking accuracy of located objects while continuing exploration.

Finally, the problem of minimum time planning under uncertain or incomplete maps is considered, and an optimal planner which guarantees collision avoidance is developed. This planner allows near real-time computation while proving smooth dynamically feasible trajectories.

BIOGRAPHICAL SKETCH

Alexander Ivanov received his B.S. in Electrical and Computer Engineering from Texas A&M University in 2012 with a focus on controls and mathematics. He received his M.S. from Cornell University in 2016 with a focus on robotic planning. His research and interests lie in optimal robotic planning and control.

I dedicate this work to my father Ivan. For being the one shred of stability in a crazy life. I could not have asked for a better guide in life. Without him, this work would never have even begun.

ACKNOWLEDGEMENTS

I'd like to thank all those who helped steer me in the right direction during my graduate studies and shone as examples and ideals. I strive to be as brilliant as they are in their own ways. The list is long, but I will mention a few here.

Raphael Luca: for keeping me sane in my first year and showing me true grit.

Nisar Ahmed: for showing me what a peer mentor should be.

Peter Radecki: for setting the highest bar in the seamless combination of practical know-how, grounded thought, and intelligence.

Zachary Manchester: for providing an intellectual ideal which is hard to match.

Paula Dobrawa: for giving an example of true delight in the work she does.

Lucas De La Garza: for struggling with me in the depths of a dark laboratory for years.

Mark Campbell: for showing me that a good advisor not only knows the subject at hand, but understands the limits of students.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Robots, either physical or virtual, have become ubiquitous in modern life, but their autonomous application has, thus far, been relegated to well known, well structured, or benign environments. Perhaps the three most un-structured applications familiar to the general public are the ®Rumba vacuum, the Mars rovers, and current efforts on autonomous cars [1]. By far the most challenging of these well known applications, in terms of path planning and control of the robots themselves, is autonomous driving.

Currently, autonomous vehicles function relatively well only in previously mapped (known) environments such that the estimation and control problems are reduced to tracking, and moving around, moving obstacles such as pedestrians, bicyclists, and other vehicles [2]. Conversely, robotic behavior in unknown environments has yet to be shown to work in any large scale application outside a research lab. As such, robotic planning and exploration in unknown environments provides a fruitful area of on-going research and development.

The vision of any autonomous research is to create robust, fully functional robots able to perform their tasks seamlessly while adapting to a complex and ever changing environment. As roboticists, we wish to cut our creations off from the proverbial apron strings, but the course of scientific progress is slow and plods along. As such, we have to settle for incremental improvements and problem simplifications which can handled concisely using the current state-of-the-art. The vision that this works seeks to address is the following. Imagine a team of autonomous vehicles which awake in an unknown place. They seek to perform some task, perhaps surveying the area, finding their missing human masters, or locating some valuable resource. The robots do not have any conception of their immediate environment and may not have direct communication with each-other. In the ideal case, the robotic agents would act as would their human counterparts and make a "robotic village." Some agents may seek out their compatriots to establish

communication while others would begin to map the environment to identify unique landmarks used for navigation and dangerous obstacles. A third group would then use this knowledge to begin to search for and execute their objective. The robotic village would have a distribution of labor and would be robust to a lack in communication, environmental knowledge, and would be able to execute their objective at speed.

This work presents three contributions in the area of robotic planning in un-known or uncertain environments. These contributions address simplified problems in the "robotic village." The first contribution focuses on guaranteed exploration and localization for ground robots in a partially known environment. This contribution's primary novelty lies in combining concepts of exploration and mapping with planning and localization. In effect, the first contributions seeks to address the goal of the mapping robots in the village. The second contribution focuses on the coordination of multiple robotic agents in an exploration and tracking task. Agents must first locate, and then provide tracking guarantees on, unknown targets of interest in a coordinated fashion. The second contribution seeks to address the issue of achieving a higher level goal, such as finding missing humans, in a coordinated way. Thus, the second group of robots are the miners or producers of the village and seek to achieve their goal cooperatively. The final contribution focuses on guaranteed minimum time planning for a robotic agent in a partially known static obstacle map. This problem generalizes the classical Jogger's Problem [3] and provides guaranteed aggressive collision-free paths even when obstacles in the environment are not fully mapped. The final contribution seeks to address robustness of individual robots when trying to achieve their objectives quickly. Conceptually, this problems seeks to make our robotic villagers less clumsy and able to "run through the forest" to attain their goals instead of moving slowly and conservatively.

NOMENCLATURE

Symbols for Uncertainty Constrained Robotic Exploration

$\beta \in (0,1)$ Scalarization parameter

$\delta, \alpha \in (0,1)$ Constraint parameters defining probability of localization

$\mathbb{E}[\cdot]$ Expectation

$\gamma \in \mathbb{R}^+$ A constraint on the covariance norm

$\mathcal{C}$ Cardinality

$\mathcal{G}(V,E)$ A graph with vertex and edge sets

$\mathcal{L} \subset \mathbb{R}^2$ Localization region

$\mathcal{N}(x,\Sigma)$ Normal distribution with mean x and covariance Σ

$\phi, \theta \in (0,1)$ Mis-detection and false alarm parameters

$\Sigma_k \in \mathbb{R}^{n_x \times n_x}$ Robot covariance at time k

B_{info} Information reward for RH-PIE

B_{pos} Robot state error cost for RH-PIE

C_i The ith 2D grid cell

$C_{\text{free}}, C_{\text{obs}}$ Free and obstacle configuration spaces

$H(\cdot)$ Entropy

i Index variable associated with position

k Index variable associated with discrete time

$M(\cdot)$ Subset of map RVs affected as a function of the argument

$M(t)$ Set of RVs representing a map at time t

n_x, n_z, n_m Dimensions of robot state, measurement, and map

$p(\cdot), P(\cdot)$ Probability density and mass

$R(\cdot)$ Reward function mapping a path to $\mathbb{R}$

T, T_1 Time horizon variables

$tr(\cdot), \lambda_1(\cdot)$ Trace and spectral norm of a matrix

v_i, e_i Indexed vertex and indexed edge

X_k Robot state random variable

$x_k, \bar{x}_k, \hat{x}_k$ True, mean, and reference robot state at k

$X_{t:T} \in \mathcal{X}_{t:T}$ A path from t to T, and its allowable set

$Z_{t:T}$ Concatenated measurements from time t to T

Synbols for Joint Exploration and Tracking

$a_j(t), w_j(t) \in \mathbb{R}^{n_a}$ Object state and process noise

$n, m \in \mathbb{N}$ Number of robots and OIs

n_s, n_d Number of GM components (mixands), for sensor and OIs

n_x, n_a Dimention of the robot and object state

n_z, n_u Dimention of measurement, and control

$O^k_{i,j} \in \{0, 1\}$ Detection/observation of object j by robot i at time k

$x_i(t), w_i(t) \in \mathbb{R}^{n_x}$ Robot state and process noise

$z_{i,j}(t), v_{i,j}(t) \in \mathbb{R}^{n_z}$ Measurement of object j by robot i and measurement noise

CHAPTER 2

UNCERTAINTY CONSTRAINED ROBOTIC EXPLORATION: AN INTEGRATED EXPLORATION PLANNER

2.1 Introduction

Exploration in sensor limited, global-information-deficient environments is necessary for mobile robots to act independently. Applications with these challenges include persistent robots new to a home or office, or robots operating in dangerous places where human participation is impossible and GPS coverage is limited. Examples of such scenarios include burning buildings and nuclear power plants. Any robot operating in these environments needs to have knowledge of its surroundings to fulfill higher level tasks.

The problem of exploring an unknown area while trying to complete a higher level mission is termed Integrated Exploration (IE) [4]. Mission success in IE is formally defined as the ability of a robot to gather sensory information about its surroundings, while maintaining pose and/or maintaining the ability to recover from pose degradation. These two conflicting goals, along with the probabilistic nature of sensing, make IE a challenging problem.

Fig. (2.1) shows a conceptual example of integrated exploration in a partially known environment. The robot initializes at the bottom left, with a goal of exploring its surroundings. In addition, the robot must maintain a consistent pose estimate along the way (i.e. not get lost). Since pose information is sparse, the robot must eventually travel to the green goal region, whose rich pose information enables the pose uncertainty to shrink to desired levels. In practice, this goal could be an *a priori* known area with well known landmarks, a distinct doorway within a building from blueprints, or WiFi/GPS coverage. It is assumed that the robot knows the location of regions which are impassible (black), contain no pose information (white), and poor pose information (purple). In the example shown in Fig. (2.1), the robot chooses between, two possible paths. Although the red path is longer and explores more area, the expected pose information is sparse. Conversely, the blue path explores less, but has much more pose information. The key research question addressed in this paper is: How can the robot optimize the exploration objective with a constraint of maintaining a globally consistent pose.

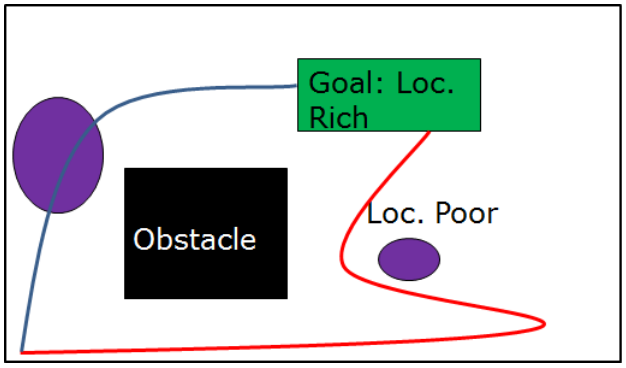


Figure 2.1: Conceptual exemplification of the partially known IE problem: The robot must navigate from the start point to the goal region, avoid obstacles, and maintain a globally consistent pose estimate. It can gather some localization information in the purple regions, and rich localization information in the goal region.

Makarenko et al. study a planner with information gain as a path reward [4]. Paths are generated towards a 'Frontier' which differentiates between visited and unvisited areas of a map. A similar approach is taken by Freda et. al. [5] where a set of randomized points is evaluated at each time step for their information and localization potential. These points serve as greedy next positions. Stachniss and Burgard explore a combination of a heuristic 'closest location,' and a location with "maximum information gain" to determine a one-step look-ahead path [6, 7]. In each of these works, information-greedy paths are generated which do not consider the long term reward of their decisions. In addition, they provide no guarantee of successful completion of either the exploration or localization goals.

Martinez-Cantin uses the Minimum Mean-Square Error (MMSE) of a robot's pose and a set of localizing features as a path cost [8]. A set of controllers is then used to execute this MMSE path. Sim and Roy consider an exploration algorithm to optimally reduce the uncertainty of landmarks through a breadth first search of a gridded map [9]. Navigation function techniques incorporate considerations such as terminal path uncertainty, distance from obstacles, and localization [10, 11, 12]. Although these solutions consider path execution, re-localization, and longer look-ahead distance than do greedy methods, they provide no guarantees on completion of paths or their exploration objective.

Lunders, Levine, and How build on the Rapidly Exploring Random Tree (RRT) algorithm to generate a chance constrained subset of paths from a start position to a goal. These paths ensure a high probability of obstacle avoidance [13]. Although collision avoidance is assured, mission goals associated with information gathering tasks are either not considered or not guaranteed [14]. Carlone and Lyons create an MPC planner in an environment with known obstacles and bounded disturbances, and provide guarantees on collision avoidance while attempting to completely explore a space in an frontier sense [15, 16]. Here, the speed of information gathering is not considered directly, which implies that the robot may take a prohibitively long time to explore the area.

In summary, current work attempts to solve IE using either unconstrained reward functions or chance constrained methods for obstacle avoidance. Unconstrained methods enable information gathering, but make no guarantees. Chance constrained methods guarantee path completion but tend to explore a space slowly [13, 14].

This work builds on the Guaranteed Probabilistic Information Explorer (G-PIE) in [17]. The G-PIE is an information theoretic path planning algorithm that provides a solution to the IE problem with the characteristics of fast exploration *and* guarantees on re-localization.

This work first defines the Integrated Exploration (IE) problem, along with the specific subproblem of information gathering in partially known environments, the focus of this paper, in Sections 2.2 and 2.3. An information reward function is introduced followed by two important novelties: a bound is derived on the proposed information reward function, and a probabilistic re-localization constraint is developed in Section 2.3.5 and 2.3.3. These components enable the key attributes of the G-PIE formulation: 1) fast exploration; 2) asymptotically optimal exploration; 3) a probabilistic guarantee of localization. Finally, expanding on [17], a receding time horizon implementation of the G-PIE is developed which addresses the exponential complexity of G-PIE, and allows near-real time planning in Section 2.4. The behavior of the receding horizon

G-PIE is analyzed via fully autonomous hardware-in-the-loop experiments in Section 2.5.

2.2 General Framework for Integrated Exploration with Chance Constraints

A framework for robotic path planning with information collection goals and probabilistic constraints is formulated with two major components: a reward function which facilitates information collection, and chance constraints which provide probabilistic guarantees on a problem-dependent goal.

In a 2D environment, the robot pose at time $t \in \mathbb{N}$, X_t, consists of the 2D coordinates of a robot and its orientation. A path starting at time t and terminating at time T is denoted $X_{t:T}$ and the optimal path, $X^*_{t:T}$, maximizes a reward, $R(X_{t:T})$:

$$X^*_{t:T} = \text{argmax}_{X_{t:T}} R(X_{t:T}), \quad R : \mathcal{X}_{t:T} \to \mathbb{R}^+. \tag{1}$$

Note that, $X_{t:T}$, is a random variable and has a distribution. In this work the terms path and path distribution are used interchangeably.

Information collection tasks, such as exploration, can often be encoded as a function of Random Variables (RVs). Thus, quickly gathering information about these variables constitutes fast exploration. Several well known metrics which capture uncertainty of RVs are Fisher Information (FI), Kullback-Leibler Divergence (KLD), and Differential Entropy (DE) [18]. When considering an n_m dimensional joint distribution, FI can be computationally expensive ($O(n_m^3)$) and require large amounts of storage ($O(n_m^2)$) [18]. KLD measures the difference between two distributions. This is not sufficient to ensure information gain, however, because, although two distributions might differ drastically before and after an observation, their overall uncertainty may be similar (e.g covariance). In contrast to KLD, entropy provides an intuitive information-

theoretic metric. A random variable has high entropy when it is very uncertain, and the entropy of a RV monotonically decreases as the RV becomes more certain. For these reasons, entropy is used as the information metric in this work.

RVs are denoted by capital variables and their realizations are lowercase. The domain of a random variable such as X_t, is denoted by capital script $\mathcal{X}_t$. Therefore, the DE of a RV, such as X_t, is defined as:

$$H(X_t) = -\int_{x \in \mathcal{X}_t} p(x) \log(p(x)) dx. \tag{2}$$

In IE, a vector valued RV can represent quantities of interest, such as uncertain target positions or obstacles. As one becomes more certain of the true value of a RV, the corresponding DE $H(\cdot) \to -\infty$ in the continuous case and 0 if the RV is discrete. For the IE problem, most variables of interest are contained in a vectorized map variable $M(t)$ which has been sensed by measurements $Z_{0:t}$. An information theoretic reward function can be defined using entropy as:

$$R(X_{t:T}) = H(M(t)) - \mathop{\mathbb{E}}_{Z_{t:T}} \left[M(T) \right]. \tag{3}$$

The above equation is termed mutual information. Here, $Z_{t:T} = \{z_t, ..., z_T\}$ is the set of measurements taken of M from time t until some terminal time T after which no further measurements are taken (i.e. path completion time). The dependence of $M(t)$ on $Z_{0:t}$ is suppressed in Eq. (3) to reduce clutter and should be read as $(M(t)|Z_{0:t})$. Eq. (3) rewards information gain.

Chance constraints are a form of soft constraint which guarantees a given condition with high probability. For IE, chance constraints are assumed to be represented by a vector function $f(\cdot)$ and a constant vector w. These constraints take the form:

$$f(X_{t:T}) \leq w. \tag{4}$$

In the case of exploration, the distributions on the random variables associated with the robot location (i.e. the path) encode many variables such as variability in $M(t)$, availability of location

information, sensor accuracy, etc. Examples of practical constraints include obstacles to avoid and uncertainty in robot location. The general framework for information theoretic goals with guarantees via chance constraints is given as Eq. (3) & (4).

2.3 Exploration in Partially Known Environments

This section provides a formulation of the Integrated Exploration problem which enables probabilistic guarantees on relocalization when obstacles in the environment are already known.

2.3.1 List of Assumptions

1 At least one region, $\mathcal{L}$,with 'good' pose information is assumed known. There are many scenarios in which such a region is known *a priori* such as when distinct doorways in a building are known from blueprints.

2 The obstacle and free spaces C_{obs} and C_{free} are known. This assumption is valid when an exploration area has been previously mapped but richer information, such as the location of structural damage around buildings, is required.

3 The space is represented as a 2D grid composed of n_m probabilistically independent grid cells C_i, a standard assumption in exploration problems [19].

4 The allowable set of reference paths are node orderings from a mathematical graph $\mathcal{G}(V, E)$, and reference paths can be well followed by a low-level controller, i.e. $\hat{x}_{t:T} \approx x_{t:T}$. This assumption has been used to great effect in recent literature such as [20].

5 The reference paths are required to be *simple*, i.e. nodes can only be visited once within a path. For information exploration problems, expected entropy is subject to diminishing

returns with more measurements (Fig. 2.5). Therefore, restricting paths to be simple is a reasonable assumption.

2.3.2 Problem Formulation

The primary focus of this paper is the IE problem in *partially known* environments. Colloquially, the robot can be seen as a 'tourist' who has a high level overview of an area, but still wants to explore this area in detail. This scenario implies knowledge of the location of impassible obstacles and localization areas and their accuracy. Localization Poor Areas (LPAs) are regions with highly uncertain localization information. Examples of LPAs include prior mapped SLAM landmarks or areas with known unique features which are difficult for the robot to detect. For example, computer vision feature detection algorithms are error prone. Location Rich Areas (LRAs) are regions with highly certain location information such as well known mapped landmarks, WiFi regions, or areas in which GPS is available. Formal definitions of these regions are given shortly.

The robot is tasked with gathering information about the location of 'areas of interest' without getting lost along the way. In a realistic scenario, the robot may be searching for wounded soldiers on the ground or potential victims at the windows of damaged buildings. The robot plans paths from its current location through the environment to search for areas of interest. The robot is constrained to terminate any planned paths inside the LRA, $\mathcal{L}$, with high probability. This requirement ensures the robot's ability to localize and continued exploration.

Given the assumptions, it follows that:

$$\mathop{\mathbb{E}}_{Z_{t:T}}[H(M(T))] = \mathop{\mathbb{E}}_{Z_{t:T}}\left[-\sum_{i=1}^{n_m}\sum_{j=0}^{1} P(C_i = j)\log(P(C_i = j))\right] \tag{5}$$

Here $M(t)$ encodes the location of areas of interest. The measurements $Z_{t:T}$ implicitly de-

pend on the path realization $x_{t:T}$, but dependence is suppressed to avoid notational clutter. An 'interesting' cell takes value $c_i = 1$, while an uninteresting cell takes value 0.

Several computational challenges arise, which are addressed by assumption 4. The distribution of the robot's path, $X_{t:T}$, is assumed to follow a deterministic reference trajectory $\hat{x}_{t:T}$ closely enough to allow a certainty equivalence approximation, $x_{t:T} \approx \hat{x}_{t:T}$ [20]. The realization of $\hat{x}_{t:T}$ is presented shortly. Without this assumption, Eq. (5) requires integration over all possible robot states $\{x : p(X_t = x) > 0\}$, which is not tractable in general.

2.3.3 Localization Constraint: Probabilistic Guarantees

The proposed information explorer must search for areas of interest over an extended time period; this implies the robot must robustly execute many paths over time. Given that the exploration region has sparse localization information, the robot must adequately recover pose as it moves about the space. A constraint is proposed to ensure that the robot is probabilistically guaranteed to recover adequate pose. A metric is first defined to determine the level of pose information in an area, which in turn enables the definition of regions of rich and poor localization information (LRA, LPA). A location is considered a rich source of localization information if a robot can reduce its pose uncertainty to an acceptable level by taking pose measurements. Formally,

Definition 2.3.1 A point x is γ *Localizable* **at time instance** k if

$$tr(\Sigma_k) < \gamma, \quad X_k = x$$

By setting γ small, the user requires a more accurate pose estimate.

Consider now a path $X_{t:T}$ with a terminal location X_T. If the robot's pose uncertainty is large

near the terminal point of the path, X_T, the robot must then localize before planning a new path. This desired behavior is described as a terminal constraint:

Definition 2.3.2 A realized point x satisfies a (γ, δ) *Localization Constraint* if

$$P\left(\min_{t\in[T,\infty]}[tr(\Sigma_t)] < \gamma | X_T = x\right) \geq 1 - \delta.$$

By setting δ small, the user specifies a tighter constraint on the certainty of satisfying the localization metric.

The set of all (γ, δ) feasible points is defined as $\mathcal{L}$, or the LRA. This set can be seen as a 're-localization' region where the robot can reduce its pose uncertainty to acceptable levels. This constraint implies that if a robot remains at position x, it has a $1-\delta$ chance of recovering adequate pose certainty, defined by γ. Given that the robot completes, and replans, many such exploratory paths during a mission, these LRAs and constraints give a formal confidence on the ability of the robot to explore over an extended period of time. In this work, $\mathcal{L}$ is assumed to be known; $\mathcal{L}$ can be calculated numerically by discretization as shown in [4]. Furthermore, it is assumed that $\mathcal{L}$ can be represented by a set of convex polygons for ease of computation. This is a mild assumption since all polygonal regions can be triangulated and sets of polygons can approximate most practical regions.

The localization constraint in Def. (2.3.2) is challenging to compute; therefore, a stricter sense of localization is considered which proves to be more manageable:

Definition 2.3.3 A point x is *Almost Surely γ Localizatiable* if

$$\lim_{t\to\infty} P\left(tr(\Sigma_t) < \gamma | X = x\right) = 1.$$

The limit in Def. (2.3.3) has been found to have a closed form solution in the EKF case [4]. Given

these notions of 'localizability', a practical and tractable *path* constraint which probabilistically guarantees localization is defined as:

Definition 2.3.4 A path, $X_{t:T}$, is *Localizably Feasible* (LF) if

$$P(X_T \in \mathcal{L}) \geq \alpha$$

This constraint assures that a robot can adequately, with user defined probability α, localize in a (γ, δ) sense upon path completion. The general problem is now given as:

$$\begin{aligned} \underset{X_{t:T} \in \mathcal{X}_{t:T},\, T}{\text{maximize}} \quad & R(X_{t:T}) \\ \text{subject to} \quad & P(X_T \in \mathcal{L}) \geq \alpha. \end{aligned} \tag{6}$$

2.3.4 Path Representations and Optimization

Given the formulation in Eq. (6), two components are required to complete the formulation: 1) the form of $\mathcal{X}_{t:T}$; 2) the optimization strategy.

Path Generation

Discrete sampling methods are used to generate reference paths for several reasons. First, sampling methods allow a detailed set of paths to be created while using a relatively small number of sample points. Second, sampling methods have *complete* variants; i.e. variants which guarantee approximating any path between two points infinitely well as the number of sample points grows [21].

A path graph, denoted $\mathcal{G}(V, E)$, is defined by a set of sampled vertices V in C_{free}, and edges, E, connecting elements of V. A reference trajectory generated from the graph $\mathcal{G}$ is piece-wise-linear, and denoted $\hat{x}_{t:T} = \{\hat{x}_{k_1}, ..., \hat{x}_{k_2}\}$, where each $\hat{x}_k \in V$ and $(\hat{x}_k, \hat{x}_{k+1}) \in E$. The trajectory of the robot $X_{t:T} \in \mathcal{X}_{t:T}$ is therefore a function of the reference trajectory $\hat{x}_{k_1:k_2}$. A variant of the Probabilistic Road Map (PRM) algorithm is used to generate the graph, $\mathcal{G}$, which represents the allowable class of piece-wise-linear reference paths. The PRM algorithm is desirable because it allows for fast computation of the proposed pose constraint given in Def. (2.3.4) [20].

LF Path Calculation

Each potential path, $X_{t:T}$, must be evaluated as to whether it is an LF path as defined in Def. (2.3.4). This is accomplished via predictive simulation. The simulated robotic system attempts to follow the reference trajectory $\hat{x}_{k_1:k_2}$ using a trajectory following controller and an EKF filter. The generated controls are used to forward-propagate the simulated robot's distribution, while expected 'average' measurements from LPAs along the reference trajectory are used to update the simulated EKF covariance matrix [19]. Finally, the terminal distribution of X_T is evaluated in reference to $\mathcal{L}$ to obtain a probability of relocalization. For further details on the simulation process, see [20]. Note that the terminal covariance associated with each path can be efficiently calculated by employing the 'one-step' covariance update matrices described in [20]. The terminal localization constraint is then written as:

$$f(X_{t:T}) = P(X_T \in \mathcal{L}) = \int\limits_{x_T \in \mathcal{L}} \frac{\exp\left((x_T - \bar{x}_T)^T \Sigma_T^{-1} (x_T - \bar{x}_T)\right)}{\sqrt{(2\pi)^{n_x} |\Sigma_T|}} dx_T. \tag{7}$$

where $\bar{x}_T$ and Σ_T are the mean and covariance of the robot pose at the end of path simulation, and n_x is the dimensionality of X. This integration may be difficult to compute for an unstructured LRA $\mathcal{L}$. However, restricting $\mathcal{L}$ to be a union of polygons, Eq. (7) can be quickly evaluated

by sampling $\mathcal{N}(\bar{x}_T, \Sigma_T)$, and checking for inclusion in $\mathcal{L}$. The probability in Eq. (7) is then approximated as the proportion of samples found inside $\mathcal{L}$. This probability can be computed to any precision desired at a cost of of the number of samples: $\mathcal{O}(n)$.

Optimality of Exploration Algorithms

The goal of the optimization in this problem is to maximize a reward not minimize a cost. Because the optimization is performed over $\mathcal{G}$, this is a longest path problem [22].

Theorem 2.3.5 *If a polynomial time approximation algorithm exists to solve the exploration problem defined by Eqs. (1, 3) over a general graph $\mathcal{G}(V, E)$ within an arbitrary constant error $\mathrm{e}_r \in \mathbb{R}$ then $\mathbf{P} = \mathbf{NP}$.*

The proof of Theorem 2.3.5, given in the appendix, shows that the maximization of Eq. (3) over a graph $\mathcal{G}$ is more general than the Hamiltonian Path problem [23]. This proof leverages the work in [22] which shows that a polynomial time algorithm which approximates the Hamiltonian Path within an arbitrary constant is equivalent to proving that $\mathbf{P} = \mathbf{NP}$. Theorem 2.3.5 can be generalized to include a multitude of exploration and minimal covariance problems, such as that studied in [20], and shows that optimality guarantees of algorithms such as that in [20] cannot be made. Theorem 2.3.5 provides theoretical limits of performance for exploration problems which rely on general graph structures produced by algorithms such as the PRM and implies that an exhaustive search must be used to maximize Eq. (3).

Optimization of the Reward

Given a set of LF feasible paths that satisfy the threshold in Def. (2.3.4), the reward function can then be optimized to find the 'best' exploration path via a graph search algorithm. A node v_{start}

is added to $\mathcal{G}$ at the robot's initial pose estimate, and a goal vertex v_{goal} is defined as the centroid of one the members of $\mathcal{L}$; optimization of the location v_{goal} within $\mathcal{L}$ is left to future work.

Due to the non-linearity of expected entropy in the reward function in Eq. (5) as well as the fact that the IE problem is maximizing a positive reward, additive graph search algorithms such as Dijkstra's and A* cannot be used [24]. This work utilizes a modified Depth-First Search (DFS) to iterate through all potential paths from v_{start} to v_{goal} to find the optimal path.

The expected entropy computation in the reward function, Eq. (5), increases intractably with the length of the measurement $Z_{t:T}$. In order to alleviate this scaling problem, the entropy of only the subset of cells which will be measured along the reference path $\hat{x}_{k_1:k_2}$ is computed. These measured cells are known due to assumption 4: $x_{t:T} \approx \hat{x}_{t:T}$. The measurements affecting a particular cell are denoted Z^{c_i}. Many of the cells in the exploration space have no relevant measurements and their expected change in entropy need not be computed, i.e.:

$$\mathop{\mathbb{E}}_{Z_{t:T}}[H(C^i(T)] = H(C^i(t)).$$

2.3.5 Reward Bound

Computation of the reward in Eq. (5) can be greatly reduced by utilizing a tight bound on the entropy of cells. In the sequel derivation a single cell is analyzed. Thus the dependence of Z on c is dropped and the measurements are assumed to start at index 1. In order to obtain a bound for Eq. (5), several assumptions are made. First, the probability of mis-detection, $1-\theta$, and false alarms, $1-\phi$, are assumed identical. Second, each measurement, Z_j, is assumed to be taken from an $i.i.d.$ mixture of Bernoulli RVs. Finally, it is assumed that the sensor sampling frequency is fast in comparison to the motion of the robot, implying that some cells C_i have a large number of samples, n_z. The Bernoulli distributions follow the probabilities:

$$\theta := P(Z_j = 1|c = 1) = P(Z_j = 0|c = 0) := \phi. \tag{8}$$

Given the stated assumptions, Prop. 2.3.6 holds.

Proposition 2.3.6 *A bound on the reward of a cell:*

$$\lim_{n\to\infty} \mathop{\mathbb{E}}_{Z_{t:T}} [H(C|\{Z_1, ..., Z_{n_z}\})] \geq \left(\frac{1}{2}\log\left(\frac{e}{2}\right)\right)$$

The proof of Prop. (2.3.6) is left to the appendix.

Note that the bound in Prop. (2.3.6) does not depend on θ, thus the assumption that $\phi = \theta$ is not restrictive. As such, the worst case (in terms of certainty) between the false alarm and misdetection rates is taken to be $(1 - \theta)$. In practice, this approximation can reduce the computation time of $R(X_{t:T})$ by orders of magnitude, which makes Prop. 2.3.6 a key result of this work. More formally, the computation required to evaluate the expected entropy of a cell traditionally takes $\mathrm{O}\left(K \times n_z\right)$ where $K \in \mathbb{R}$ [18]. Proposition (2.3.6) makes this computation a simple subtraction ($\mathrm{O}\left(1\right)$). In MATLAB, this difference in computation time is on the order of 10^3 for $n_z = 100$.

Using Prop. 2.3.6 the measured space is divided into two parts: that with many measurements M_{cert} , and that with few M_{uncert}. Clearly $M(T) = M_{\text{cert}}(T) \cup M_{\text{uncert}}(T)$ and $M_{\text{uncert}} \cap M_{\text{cert}} = \emptyset$ (i.e. these are 'certain' and 'uncertain' parts of the space).

$$\sum_{C_i \in M_{\text{cert}}} \mathop{\mathbb{E}}_{Z^{c_i}_{t:T}} [H(C_i)] \geq \mathcal{C}(M_{\text{cert}}) \cdot \left(\frac{1}{2}\log\left(\frac{e}{2}\right)\right) := -\bar{R}_{\text{cert}}.$$

In practical scenarios such as those in the results section, a majority of the measured cells in $M(T)$ will also be in M_{cert}(T), hence Prop. 2.3.6 significantly increases evaluation of Eq. (5). The reward in Eq. (5) is bounded by:

$$
\begin{aligned}
R(X_{t:T}) \geq \bar{R}(X_{t:T}) := \\
H(M(T)) - \sum_{C_i \in M_{\text{uncert}}} \mathop{\mathbb{E}}_{Z^{c_i}_{t:T}} [H(C_i)] + \bar{R}_{\text{cert}}.
\end{aligned} \tag{9}
$$

2.3.6 Summary: The G-PIE Algorithm

An exploration algorithm with localization constrains can now be fully described by combining the reward function in Eq. (5), the constraint defined in Def. (2.3.4), and Depth First Search (DFS): (Alg. 1).

1 $R_{\text{best}} = 0$;
2 $X_{\text{best}} = \emptyset$;
3 $\mathcal{G}(V, E) = PRM(C_{\text{free}}, C_{\text{obst}})$;
4 $V \leftarrow Add\Big(v_{\text{start}} = \hat{x}_0\Big)$;
5 $V \leftarrow Add\Big(v_{\text{goal}} = x_{\text{goal}}\Big)$;
6 **for** $\big(X_{t:T} | \{\bar{x}_0 = v_{\text{start}}, \bar{x}_T = v_{\text{goal}}\}\big)$ **do**
7 **if** $P(X_T \in \mathcal{L}) \geq \alpha$ **then**
8 $Compute : \bar{R}(X_{t:T})$;
9 **if** $\bar{R}(X_{t:T}) > R_{\text{best}}$ **then**
10 $R_{\text{best}} = \bar{R}(X_{t:T})$
11 $X_{\text{best}} = X_{t:T}$
12 **end**
13 **end**
14 **end**
15 **return** $(X_{\text{best}}, C_{\text{best}})$

Algorithm 1: The Guaranteed Probabilistic Information Explorer (G-PIE) Algorithm. Depth First Search is used in line 6.

The Guaranteed Probabilistic Information Explorer (G-PIE) algorithm behaves as follows. First, the best reward and best path, R_{best} and X_{best}, are reset, and a PRM graph, $\mathcal{G}$, is generated. Next, the start and goal nodes are added to $\mathcal{G}$, and each potential path from v_{start} to v_{goal} is checked for Localization Feasibility (LF). If a path, $X_{t:T}$, is feasible, its bounded reward, $\bar{R}(X_{t:T})$, is computed and checked against R_{best}. If required, the best path and best reward are then up-

dated, and the process repeats until all paths have been calculated or an allotted computational time has expired. Given sufficient computational time, the algorithm is guaranteed to return the optimal LF path within the current graph $\mathcal{G}$. In addition when using a complete variant of the PRM, the path found by G-PIE converges to the true optimal as $\mathcal{C}(V) \to \infty$ [21].

2.4 A Receding Horizon G-PIE

The G-PIE algorithm is guaranteed to provide optimal exploratory trajectories. However, G-PIE relies on an exhaustive search, creating computational challenges, and does not take into consideration uncertainty of the robot's intermediate pose in the reward function. Theorem (2.3.5), shows that on a general graph structure, no exploration algorithms can guarantee being within an arbitrary constant of optimal. In addition, imposing an overly restrictive structure, such as a tree, on the $\mathcal{G}$ makes it poorly approximate optimal trajectories. These facts imply that any approximation algorithm should be focused on finding local optima.

To address computational scaling, a receding horizon can be utilized. Note that as the horizon $T_1 \in \{1, 2, 3, ...\}$ increases, this receding horizon approach will obtain the optimal path given by the G-PIE. Furthermore, a heuristic tail cost function is presented which balances information gain and pose uncertainty dynamically beyond the horizon T_1. This tail cost function provides better sub-optimal solutions and can be dynamically tuned to seek more conservative (LF) or exploratory paths.

2.4.1 An Augmented Reward Function

The most common form of receding horizon control considers the optimization of a discrete time system over a finite horizon [25]. In this paper, the reward function is optimized over a finite

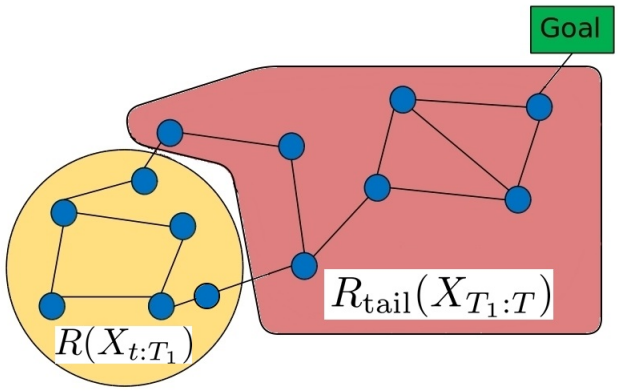


Figure 2.2: An exemplification of the horizon $T_1 = 2$. Local paths are at most two nodes away from the green start node.

number of node visitations in the graph $\mathcal{G}$; exemplified in Fig. (2.2). This is intuitive because the more nodes a robot visits, the longer its trajectory and the associated time to complete this trajectory.

The receding horizon reward function, $R_{\text{rh}}(X_{t:T})$, is split into a local reward, $R(X_{t:T_1})$, and an estimated tail reward $R_{\text{tail}}(X_{T_1:T})$. The use of $(t : T_1)$ is an abuse of notation since T_1 denotes an integer number of node visitations, as shown in Fig. (2.2), while t and T are discrete time instances.

$$R_{\text{rh}}(X_{t:T}) = R(X_{t:T_1}) + R_{\text{tail}}(X_{T_1:T}) \tag{10}$$

If an exact estimate of $R_{\text{tail}}(X_{T_1:T})$ were available *a priori*, the optimal initial path, $X_{t:T_1}$, is recovered. In simple receding horizon control problems $R_{\text{tail}}(\centerdot)$ is assumed to be zero or a large over estimate [25], but the appropriate selection of this tail reward function can lead to optimal or near optimal solutions [26].

The constraint in Def. (2.3.4) must still be enforced by any returned path maximizing $R_{\text{rh}}(\centerdot)$. Thus to ensure feasibility, it is prudent to consider a tail reward estimate which balances information gain and growth in pose uncertainty. To do this, the tail reward is defined as a scalarization of estimates of information gain, $B_{\text{info}}(X_{T_1:T})$, and feasibility, $B_{\text{pos}}(X_{T_1:T})$, with the scalarization parameter $\beta \in (0, 1)$

$$R_{\text{tail}}(X_{T_1:T}) = \beta * B_{\text{info}}(X_{T_1:T}) - (1-\beta)B_{\text{pos}}(X_{T_1:T}) \tag{11}$$

Here, $B_{\text{pos}}(\centerdot)$ and $B_{\text{info}}(\centerdot)$ are positive valued functions. The derivation of $B_{\text{pos}}(\centerdot)$ and $B_{\text{info}}(\centerdot)$ is presented next. Intuitively, when $\beta = 0$, the returned path strives to make the constraint in Def. (2.3.4) as non-binding as possible and attempts to ensure feasibility, but the optimization ignores the information content of $X_{T_1:T}$. Conversely, when $\beta = 1$, the optimizing path is more information rich, but may not satisfy Def. (2.3.4). Since Def. (2.3.4) must be satisfied, this constraint is checked for all solutions and any solution not satisfying (2.3.4) is discarded. A similar formulation in a differing application is presented in [27].

The primary purpose of the cost in Eq. (11) is to find locally optimal yet *feasible* tail trajectories. Thus, a small value of β may be required to ensure the satisfaction of the probabilistic constraint. Conversely, it is desirable to set β as large as possible to better approximate the true tail reward, and achieve better global performance. The trade between these goals, as β varies, is studied in the results.

Using Eq. (11) implies the optimization of $X_{T_1:T}$ is a shortest path problem. This is of vital importance because there are known algorithms which can solve this problem quickly, i.e. Dijkstra's. A judicious choice of $B_{\text{pos}}(\centerdot)$, $B_{\text{info}}(\centerdot)$, and β is required to guarantee a path graph with positive edge weights which allows for the optimal use such powerful algorithms.

To briefly summarize, the main goal in deriving R_{tail} is to achieve additivity and path independence in the tail sub-problem. This allows the use of shortest path algorithms to calculate R_{tail} and provides a large computational improvement over the full G-PIE solution. In particular, if the graph $\mathcal{G}$ has a maximum connectivity κ and node count ψ, then the number of paths from one node to another node in $\mathcal{G}$ is at most $\mathrm{O}\left(\kappa^{\psi}\right)$, and each of these must be evaluated by the G-PIE. In contrast, the RH-PIE complexity becomes $\mathrm{O}\left(\kappa^{T_1} * \psi \log \psi\right)$, since each of the local $\mathrm{O}\left(\kappa^{T_1}\right)$ paths must be completed by using a Dijkstra like algorithm to calculate R_{tail}. In addition,

any path which maximizes R_{rh} in Eq. (10) must still satisfy the re-localization constraint. The two different terms in R_{rh} seek a compromise between safety, which must always be guaranteed, and information, whose global optimality is sacrificed by this approximate reward.

2.4.2 Derivation of Uncertainty Penalty

The uncertainty penalty, $B_{\text{pos}}(\cdot)$, penalizes growth in positional uncertainty and is the mechanism which enables the RH-PIE to find a feasible tail trajectory $X_{T_1:T}$. The penalty, $B_{\text{pos}}(\cdot)$, in Eq. (11) must associate an uncertainty cost to traversing edges along the graph $\mathcal{G}$, and this penalty must be additive. Second, B_{pos} must be path independent to allow use of shortest path algorithms. This means traversing an edge of $\mathcal{G}$ must incur the same penalty regardless of previous node visitations. In other words, the argument of B_{pos} reduces to a node ordering. In the case of robotic motion, the pose uncertainty of the robot is path dependent; the Curse of History [28]. Instead, this work uses a bound on the maximum growth in uncertainty when traversing an edge.

In [20] the authors assume an EKF is used for robot pose estimation. They show that, along a particular edge $e_i = (\hat{x}_i, \hat{x}_{i+1})$, the update equations can be simplified by using aggregate matrices; the observation matrix $H^T Q^{-1} H \in \mathbb{R}^{n_x \times n_x}$, the noise matrix $L \in \mathbb{R}^{n_x \times n_x}$, and the propagation matrix $G \in \mathbb{R}^{n_x \times n_x}$. A detailed definition of these matrices cannot be given here and is given in [20], while details on the EKF formulation are given in [19]. The update equation is now written as:

$$\Sigma_{i+1} = L + G(\Sigma_i^{-1} + H^T Q^{-1} H)^{-1} G^T. \tag{12}$$

note that i is a spacial rather than time index. This equation implies the system is observable (perhaps weakly) over an edge. Regardless of observability, the following analysis is still valid. The matrix G is assumed to be invertible which is true in most problems [20].

Proposition 2.4.1 *Suppose, $H^T Q^{-1} H$ and Σ_i^{-1} are positive definite Hermitian matrices, then $\lambda_1(\Sigma_{i+1})$ is bounded by:*

$$\lambda_1\Big(\Sigma_{i+1}\Big) \leq \lambda_1\Big(L\Big) + \min\Big\{\lambda_1(G\Sigma_i G^T), \lambda_1(G(H^T Q^{-1} H)^{-1} G^T)\Big\} = \lambda_{\text{bound}}.$$

Corollary 2.4.2 *Using Prop. (2.4.1) the following holds:*

$$\begin{aligned} \Delta\Sigma(e_i) &:= \Sigma_{i+1} - \Sigma_i \\ \lambda_1(\Delta\Sigma(e_i)) &\leq \lambda_{\text{bound}} - \lambda_1(\Sigma_i) \end{aligned} \tag{13}$$

The proof of Prop. (2.4.1) is given in the appendix. A bound on the change in the largest eigenvalue of the robot's covariance matrix also bounds the trace of the covariance. Notice that this bound is still dependent on Σ_i. To make this bound path independent, notice that the argument inside $\min\{\cdot, \cdot\}$ which is dependent on Σ_i can be ignored and the inequality still holds. In other words:

$$\min\Big\{\lambda_1(G\Sigma_i G^T), \lambda_1(G(H^T Q^{-1} H)^{-1} G^T)\Big\} \leq \lambda_1(G(H^T Q^{-1} H)^{-1} G^T). \tag{14}$$

The bound in Eq. (13) depends on Σ_i through λ_{bound} and the subtraction of $\lambda_1(\Sigma_i)$. To eliminate this dependence, both of these terms must be addressed. Starting with λ_{bound}, an analysis of Prop. (2.4.1) shows that the first term in the minimum takes into consideration the dynamics of the robot while the second term becomes infinite as the system becomes unobservable. Many authors assume the integral observability of the system [20]. Regardless, in scenarios where no pose information is available to the robot, it is practically necessary to maintain both terms in the minimum. One approach is to consider a worst case Σ_{worst} which has only one eigenvalue (i.e. symmetric uncertainty); the worst case could be the divergence of the pose estimate. Once Σ_{worst} is identified, this value can be used for all edges of $\mathcal{G}$ in lieu of Σ_i in Prop. (2.4.1).

Thus, λ_{bound} has been made path independent, but Eq. (13) still depends on Σ_i. A looser approximation which requires no further assumptions is bounding Eq. (13) from above by λ_{bound} alone. This bound can be too loose, and it is therefore practical and convenient to define Σ_{best}. In other words, defining Σ_{best} to be such that Σ_k has a smaller spectral norm than Σ_{best} in all practical scenarios. Σ_{best} should be small enough that this level of uncertainty in position has little bearing on the performance of the robot in its mission. In this case, the most natural definition of $\lambda_1(\Sigma_{\text{best}})$ is γ/n_x, where γ is taken from Def. (2.3.1).

Now that the bound in Eq. 13 is established, the pose uncertainty penalty associated with an edge is:

$$B_{\text{pos}}(X_{T_1:T}) = \sum_{i=T_1} \lambda_1(\Delta\Sigma(e_i)). \tag{15}$$

Note that $B_{\text{pos}}(\centerdot)$ is positive because the matrices involved are positive definite Hermitian.

Interpreting this bound intuitively, notice that L in Prop. (2.4.1) represents the uncertainty added due to process noise, $G\Sigma_k G^T$ is a transformation and scaling due to robot motion, and $G(H^T Q^{-1} H)^{-1} G^T$ is the net effect of expected measurements. Thus, B_{pos} increases due to robot motion and length of the reference path, and decreases with good pose observations along the path.

2.4.3 Derivation of Tail Information Reward

In order for R_{tail} to be fully path independent, B_{info} must also be made path independent and additive. In reality, an edge e_i can give more or less information based on how the robot moved prior to traversing e_i. Thus, an approximation must be made in order to achieve additivity and path independence.

The key challenge in the approximation of information gain is due to information 'overlap'. If the robot traverses e_i before e_j more information is gathered along e_i in comparison to the case when the robot first traverses e_j then e_i. An approximation is presented here which works well in practice, and is used in the experimental section of this work. There is a variety of different approaches to make such an approximation, and the appendix provides a more detailed discussion. The fundamental idea behind this approximation is dividing the cells of the exploration space into regions where each edge e_i has an associated set of cells. The cells associated with e_i are then assumed to be independent of traversal along e_j. This assumption breaks the path dependence while also drastically improving computation speed.

This approach and assumes that any cells within sensor range of e_i are not affected by the traversal of any other edge. This implies that even if a cell c_j is within sensor range of two or more edges, the reward for traversing e_i is the same as if none of the cells within its range are observed before traveling along e_i. Let $M(e_i)$ be the portion of the exploration space visible by traversing edge e_i, then the tail information reward can be expressed as:

$$B_{\text{info}}(X_{T_1:T}) = \sum_{e_i \in X_{T_1:T}} \Big[H(M(e_i)) - \mathbb{E}_{Z_{e_i}}[H(M(e_i))] \Big]. \tag{16}$$

This value is really an over bound of the expected information gain along e_i. This method is fast to compute, and gives a reasonable estimate when the optimal path does not traverse the same area many times. An underbound and 'average' approximation of B_{info} are developed in the appendix.

2.4.4 The importance of β

Given the definitions of B_{pos} and B_{info}, the selection of β must be considered. The form of R_{tail} shown in Eq. (11) transforms the problem of finding $X_{T_1:T}$ into a shortest path problem if β is

selected such that there are no negative loops in the graph $\mathcal{G}$. Since B_{pos} is strictly positive and B_{info} is non negative, such a β can always be found. In addition, β controls how much weight is given to information gain versus maintaining low pose uncertainty.

Recall that, in an un-directed graph $\mathcal{G}$, any negative edge results in a negative loop between two nodes. Thus, β must be set to ensure non-negative edge weights throughout the entire graph, which in turn enables shortest path algorithms to return an optimal path. Although this requirement is sometimes ignored, as in [27], it is vital to ensure optimality in the tail sub-problem. The maximal value of β which guarantees positive edge weights can be computed by analyzing each edge. Even when β is larger than the value which ensures non-negativity, Dijkstra-like algorithms can return sub optimal paths. If time allows, a binary search can be used to find the largest β value which returns a feasible, more information rich, path. Once β is set $X_{T_1:T}$ can be determined. The Receding Horizon Probabilistic Information Explorer (RH-PIE) can be fully described in Alg.(2).

$R_{\text{best}} = 0$;
$X_{\text{best}} = \emptyset$;
$\mathcal{G}(V, E) = PRM(C_{\text{free}}, m)$;
$Add\Big(v_{\text{start}} = \hat{x}_0\Big)$;
$Add\Big(v_{\text{goal}} = x_{\text{goal}}\Big)$;
$\beta = \beta_{\text{user}}$
for $\big(X_{0:T_1} \in \mathcal{X}_{0:T_1} | \{\bar{x}_0 = v_{\text{start}}\}\big)$ **do**
    $[X_{T_1:T}, R_{\text{tail}}(X_{T_1:T})] = ShortestPath(\hat{x}_{T_1}, \bar{x}_T)$
    if $P(X_T \in \mathcal{L}) \geq \alpha$ **then**
        $Compute : R_{\text{rh}}(X_{0:T}) = R(X_{0:T_1}) + R_{\text{tail}}(X_{T_1:T})$;
        if $R_{\text{rh}}(X_{0:T}) > R_{\text{best}}$ **then**
            $R_{\text{best}} = R_{\text{rh}}(X_{0:T})$
            $X_{\text{best}} = X_{0:T}$
        end
    end
end
return $(X_{\text{best}}, R_{\text{best}})$

Algorithm 2: RH-PIE

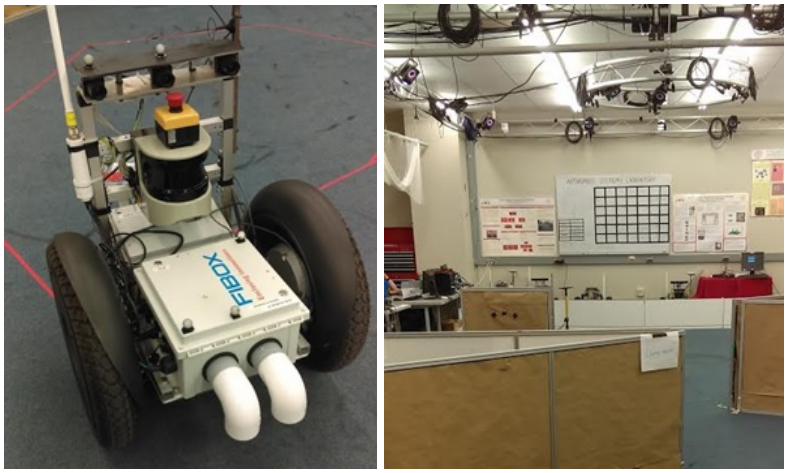


Figure 2.3: Shown on the left is the Segway RMP50 robotic platform. It has GPS, odometry, LIDAR (LMS511), IMU, and camera sensing capabilities. The right image shows the experimental obstacles and VICON positioning system used to provide ground truth.

The RH-PIE algorithm is similar in structure to the G-PIE algorithm. The primary difference is that, R_{rh} is calculated for the set of local paths within the user defined horizon, T_1.

2.5 Simulation and Experimental Results

In order to fully understand both the theoretical and practical behavior of the G-PIE and RH-PIE algorithms, two sets of results are presented.

2.5.1 Models Sensors and Experimental Setup

In both the simulation and experimental results, the robot is modeled as a unicycle with direct velocity/turn-rate control (v, ω) [19]. In order to match the modeling assumptions made between the simulations and experiments, a discrete-time, first order, linear, input-output dynamics model was fit to data from the Segway platform. A kinematic state feedback controller is used for trajectory generation and tracking [29]. This model is also used to inform the predictive step in the RH-PIE algorithm as well as in an EKF filter, which provides pose estimation.

Location measurements are noisy relative range and bearing to known landmarks (implicitly generating LPAs). Areas of interest are represented as a grid, and noisy binary measurements (interesting, uninteresting) are taken at 10Hz. In these scenarios, the edges of objects are taken to be 'interesting'. Thus, in the experiments, measurements are returns from a SICK LMS511 LIDAR. The measurements' speed implies enough information measurements can be expected to form a large subset of cells in M_{cert} in Eq. (9), assuming a robot speed of 1-3 m/s. The high number of measurements implies Prop. 2.3.6 is instrumental in accelerating the computation of the path reward by replacing Eq. (5) with Eq. (9).

In the experiments, ground truth of the robot pose is obtained using a VICON motion capture camera suite. In addition, VICON also allows the generation of software-based point landmarks, which are the basis of relative range and relative bearing pose measurements. Independent white Gaussian noise is added to each measurement. In this way, the measurement within LPAs can be precisely controlled and matched to modeling assumptions. The SICK has a 190°field of view and its range is restricted to 2m due to the constrained laboratory environment. Note that the robot does not know *a priori* that obstacle boundaries are interesting.

Comparing the experimental and simulation setups, modeling errors in robot motion, as well as non-whiteness and non-Gaussianity of the realized noise are the only modeling differences. The practical differences between experimental and simulation settings include the necessary use of a reactionary obstacle avoidance procedure, and imperfectly synchronized measurements of the environment and robot position.

2.5.2 Simulation Results

Three sets of simulations are presented to demonstrate the effectiveness of the G-PIE algorithm, verify claims, and validate assumptions. First, a qualitative discussion of the G-PIE algorithm's

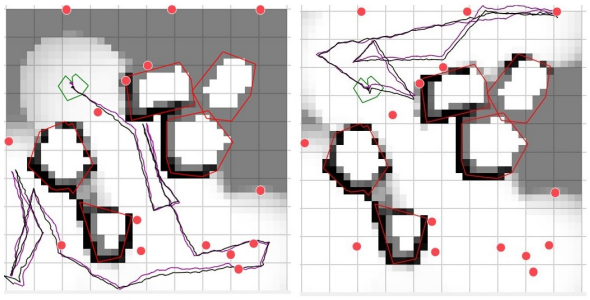


Figure 2.4: The robot starts at the bottom left with no information about the areas of interest and explores while traveling to $\mathcal{L}$. Landmarks are denoted in red, and $\mathcal{L}$ is composed of a single LRA in green. The robot's estimated EKF path is in black while its realized path is in purple. After a G-PIE path, the robot has discovered some areas of interest but a large area remains unexplored. After a second G-PIE path, the robot has discovered almost all areas of interest with high confidence.

behavior is given. This demonstrates its functionality and its ability to repeatedly plan paths that enable long term autonomy. Second, the convergence of expected entropy to the bound in Prop. 2.3.6 is analyzed and the bound's assumptions are scrutinized. Finally, Monte Carlo simulation results are shown comparing expected and realized constraint satisfaction and information gain.

Figure (2.4) shows an instance of two iterations of the G-PIE algorithm, exemplifying the trade between information gain and localization. Here, areas of interest are assumed to be the edges of objects, but the robot does not know this *a priori* and cannot sense through obstacles. The discovered areas of interest are colored in black, while still unknown regions are in gray. The robot starts in the bottom left corner of the map. In the first iteration, Fig. (2.4) left, the G-PIE algorithm selects the path with the highest possible information gain while guaranteeing a success of 95% probability of terminating in $\mathcal{L}$. The robot navigates the environment, collecting information about areas of interest. The robot then terminates within the LRA, and its pose estimate becomes more confident due to location rich measurements. The updated pose enables the robot to continue to explore as shown in Fig. (2.4) right, where the G-PIE algorithm replans from its current location back to $\mathcal{L}$. After two iterations the estimate of the regions of interest in Fig. (2.4) right is produced. In addition, the user defined threshold of $\alpha = .95$ ensures that each iteration has a 95% confidence of path completion. The remaining unexplored area is too risky, primarily because there are no nearby landmarks with which to localize.

Because $\mathcal{L}$ is only one area, the G-PIE algorithm generates paths where the robot moves around unknown areas and loops back to the same region ($\mathcal{L}$). In the case where $\mathcal{L}$ is composed of multiple LRAs, the selection of the appropriate LRA can be added to the optimization.

Convergence of the Expected Entropy

In the development of Prop. 2.3.6, the mis-detection and false alarm rates, $(1-\phi)$ and $(1-\theta)$, are assumed to be identical; this section studies this assumption and the applicability of the entropy bound in Eq. (9). Figure (2.5) plots $p(c=1|Z^c_{t:T})$ from the bound in Eq. (2.3.6) versus number of expected measurements, n, of that environment cell. A total of 10,000 Monte-Carlo cases were run using pseudo random numbers to generate samples of measurements from $p(Z^c|c=1)$ for a 3×3 test set of three cases of mis-detection/false alarm: $\theta = \phi$; $\theta = \phi + .1$; $\theta = \phi - .1$. Three cases of θ are considered, and the prior distribution of a cell being interesting is assumed to be uniform 50%. Figure (2.5) shows the symmetric case $(\theta = \phi)$ converges within 15 samples for mis-detection detection rates of 25% $(\theta = 75\%)$, which is representative of sensors in real operating environments [30]. This result is equivalent to 1.5 seconds of observation with sensors operating at 10Hz, which is realistic in practice. At $\theta = 55\%$, the convergence is much slower (> 200 samples). At this rate of false alarms, the sensor returns are incorrect 45% of the time (nearly a coin flip). Regardless, the bound is still met near 400 samples (not shown due to scale), or 40 seconds of observation at 10Hz. Thus, the robot must be in sight of a particular grid cell for reasonable period, even with a poor sensor, for Prop. 2.3.6 to be valid.

Figure (2.5) shows that, for the non-symmetric cases $(\theta \neq \phi)$, convergence is faster than the worst case scenario of $\theta = \phi \in [.55, .65, .75]$, respectively. This convergence trend is consistent for any values of θ and ϕ. As a result, Fig. (2.5) verifies that the assumption $\theta = \phi$ is conservative.

The G-PIE must evaluate hundreds or thousands of potential paths. Thus, the computation of the reward becomes significant. In MATLAB on an I5 Intel processor, computing the expected

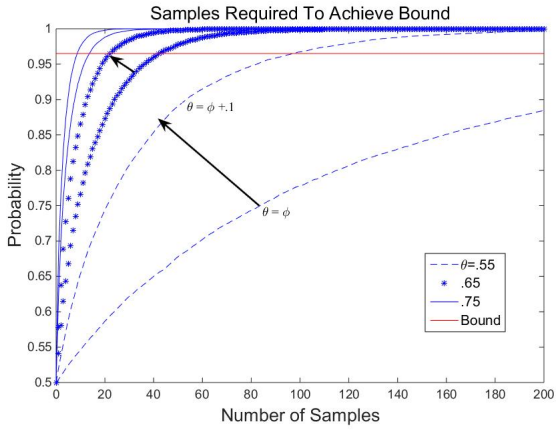


Figure 2.5: Expected probability of interest, $p(C_i = 1|Z_{1:n})$,of a cell as a function of the number of expected samples ,n, for three values of θ. Curves with corresponding styles show the difference between the cases where $\theta = \phi$ and $\theta = \phi + .1$.

entropy of 1000 cells with 30 expected samples takes $\sim$ 1sec of computation. Using Prop. 2.3.6 instead requires 10^{-3} times less computation. The study shown in Fig. (2.5) gives a guide for the number of samples required to use Prop. 2.3.6. The number of required samples is the crossing point between the expected entropy of a cell and the derived bound for a given θ and ϕ. Any number samples greater than this crossing point assures that Eq. (9) is valid.

Difference in Expected Path Completion and Exploration v.s. Realization

A Monte Carlo (MC) study is performed to evaluate Desired versus Predicted versus Realized achievement of the G-PIE planner. The Desired Achievement is simply α (Def. 2.3.4), and determines the ability of the robot to terminate inside $\mathcal{L}$. The Predicted Achievement is the probability of paths returned by the G-PIE algorithm terminating inside $\mathcal{L}$: this must always be above α by construction. The Realized Achievement is the proportion of MC simulated runs which terminated inside $\mathcal{L}$. Obstacles, LPA, and LRA regions are randomly generated at each run. A failure is a path which does not terminate inside $\mathcal{L}$, or a collision with obstacles due to poor localization. By varying the given threshold α from .5 to .9, Fig. (2.6) is obtained. Fig. (2.6) plots Predicted (solid blue), and Realized (dashed red) Achievement as a function of Desired Achievement (solid,shaded black). Each data point corresponds to 10,000 MC runs. While α

varies from .5 to .9, the MC Predicted Achievement correspondingly varies from .86 to .94. This implies that the robot is able to find informative paths that also have a high probability of termination in $\mathcal{L}$ (i.e. the Predicted Achievement $>> \alpha$). The Realized Achievement lags behind Predicted Achievement by approximately 2% in each case due to obstacle collisions and nonlinearity of the robotic system. Note that the G-PIE algorithm always ensures a higher path completion/localization than the given threshold α (i.e. G-PIE is conservative from the user's perspective).

2.5.3 Experimental Results of RH-PIE

To verify the practicality of the RH-PIE algorithm and analyze the effects of algorithmic parameters and environmental complexity, several sets of hardware experiments are presented. In this section, three key questions are addressed: 1) What effect does the choice of β have on the solutions generated by the RH-PIE algorithm and how restrictive is the positive edge weight requirement for tail cost optimality? 2) What is the effect of the theoretical assumptions, particularly environmental complexity in the RH-PIE algorithm's development? 3) What effect does the look ahead distance have on path quality, optimality, and computational complexity?

Three distinct maps are used, which enabled the study of key parameters independently: obstacle complexity, initial position, look ahead distance, and β. In each map, two LRAs are provided in the same locations, shown in Fig. (2.8) as green polygons. The map is 3.5m $\times$ 9.5m and has six localization landmarks as seen in Fig. (2.8). It is important to note that each trial utilized the same underlying graph, $\mathcal{G}$, of 80 nodes to plan over. The graph is generated with a minimum connection distance of 0.5m and a maximum connection distance of 1m. This implies that a look ahead of 1 is between 0.5m and 1m. The location and quantity of positional landmarks remains the same between maps. In terms of obstacles, the first map contains no obstacles, the second map contains one large central obstacle, and the third map contains three obstacles. Finally, to

ensure repeatability, an automated initialization procedure was implemented to ensure the robot began each trial within 5cm and 1° of its intended initial condition (IC).

The effect of β

Recall that when β is near 0, the RH-PIE algorithm seeks only conservative information gathering paths which maintain an accurate pose estimate away from the local neighborhood of the robot. Conversely, setting β near 1 should cause the robot to exhibit more exploratory behavior. Regardless, the RH-PIE algorithm must satisfy the localization constraint and thus must guarantee relocalizing with high probability.

To evaluate the real effect of the β parameter on R_{tail}, trials were run on the three proposed maps using three different initial conditions (ICs). The parameter β is swept from .1 to .9 in increments of .2. Thirty trials per map are performed for a total of 90 trials. The only variables modified are the IC and β. The robot was given a myopic look-ahead distance of $T_1 = 1$, 0.5-1m, while the feasibility threshold was maintained at $\alpha = .95$. It is noted that when $\beta > .2$, positive edge weights on $\mathcal{G}$ are not maintained and the path returned is not guaranteed to be optimal in terms of R_{tail}. Despite the loss of tail optimality, the variation in β above the .2 threshold yields qualitatively interesting and intuitive behavior.

Fig. (2.7) plots Expected Entropy Reduction (information gain) as a function of β. Clearly, as β increases the expected entropy reduction produced by X_{best} trends upward. This is because more weight is given to exploratory behavior. At values of β which ensure tail reward optimality ($\beta \leq .2$), the algorithm is relatively insensitive to changes in β. For very small values of β, the RH-PIE algorithm exhibits localization seeking behavior and strives to observe as many landmarks as possible at the expense of distance traveled. Note the large jump in expected entropy reduction between $\beta = .5$ and $\beta = .7$. This is due to local edge weights around the robot's IC becoming negative and allowing more global exploration. Initial Condition 2 has the largest jump

because the robot starts off near the goal, and the myopic look-ahead of 1 does not allow much exploratory action. Thus the robot must rely almost entirely on β to allow it to explore away from the goal location.

Fig. (2.8) shows two different planned paths on the obstacle free map for $\beta = .1$ and $\beta = .7$. The robot begins in the top right corner of the map near an LRA and plans using the RH-PIE. The short look ahead helps to reveal the effects of β more explicitly. The robot plans to the bottom left LRA while exploring. The absence of the EKF estimate implies that the EKF estimate and truth are nearly identical. Note the prior distribution imposed on areas of interest: the top left corner of the map is known to contain nothing of interest *a priori*. Thus, the only reason that the robot should traverse this region while planning a path to the bottom left LRA is to observe the cluster of positional landmarks (black dots). It is evident that the robot exhibits strong localization seeking behavior when $\beta = .1$, and chooses to pass through the uninteresting area to see the positional landmarks. In contrast, when $\beta = .7$ the robot circles the two intermediate localization landmarks to ensure a feasible trajectory, but creates a highly exploratory trajectory

It is also important to note how environmental complexity and β interplay. Figure (2.7) shows that the trends in entropy reduction remain relatively unchanged as the number of obstacles grows. However there are some differences in expected entropy reduction which result from the fact that obstacles generate different homotopy classes. This difference is especially evident when the exploration space is limited to 'corridors' generated by obstacles, as is the case in the three obstacle map. In Fig. (2.7a) and Fig. (2.7c) for IC3, such an effect is clear. In the three obstacle case, the robot is forced into an exceptionally exploratory corridor even for small β. Thus, expected information gain is higher in the three obstacle case as opposed to the obstacle free case for $\beta = 0.1$, even though the obstacle free case allows for more path flexibility. Conversely, the trend for IC3 in the three obstacle map remains relatively flat because the robot has little else explore in its limited map.

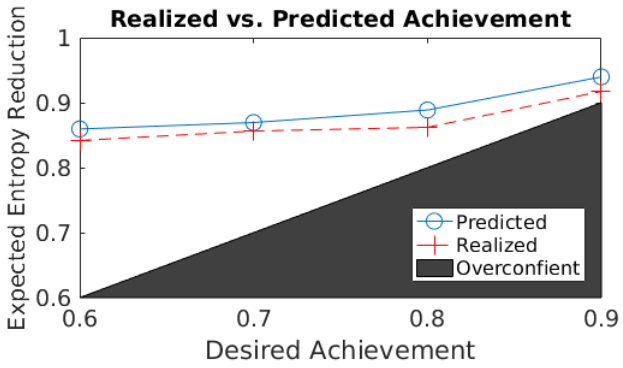
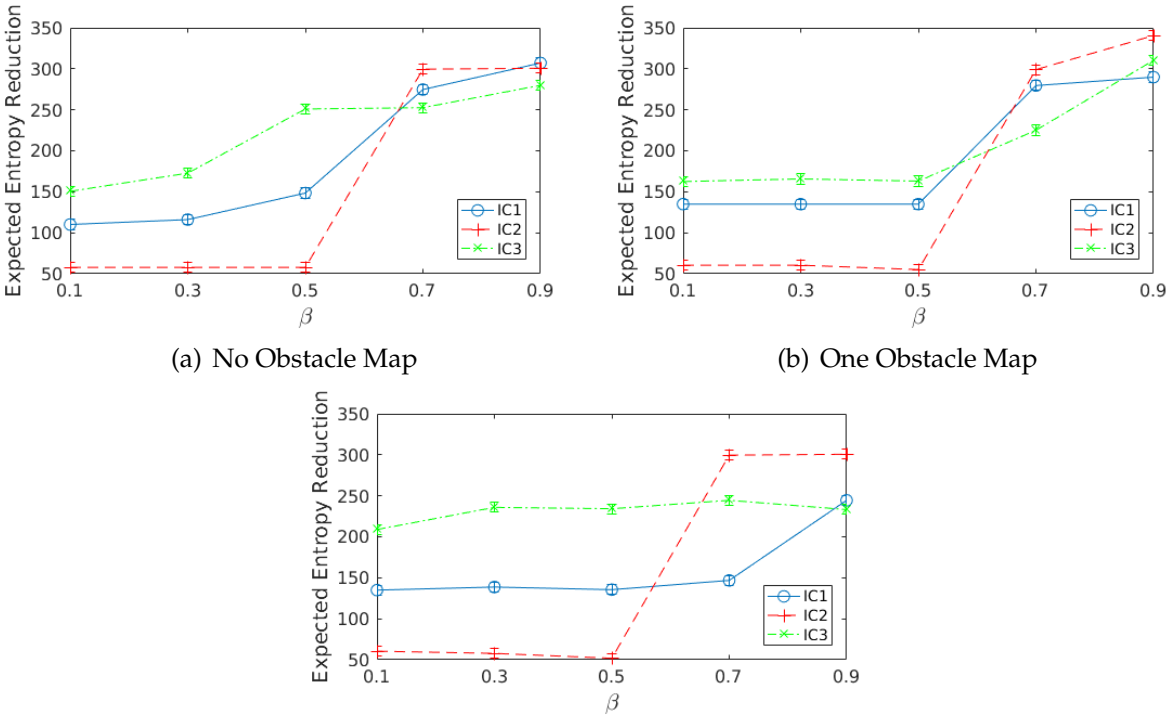


Figure 2.6: The predicted success rate for goal achievement as compared to simulated Monte Carlo realizations.

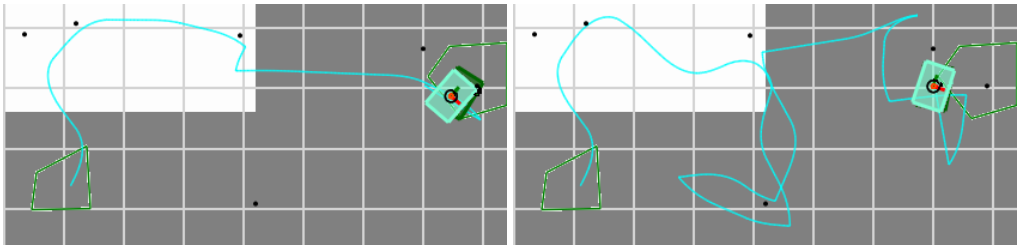


(a) No Obstacle Map

(b) One Obstacle Map

(c) Three Obstacle Map

Figure 2.7: The expected information gain as a function of β for three environments and three ICs. For $\beta < .02$, the $R_{\text{tail}}(.)$ is guaranteed to be optimal. Any returned path satisfies $\alpha = .95$. The maximum error due to IC variation is denoted by error bars.



(a) $\beta = 0.1$ (b) $\beta = 0.7$

Figure 2.8: Two RH-PIE paths for two values of β. The robot is attempting to traverse the cyan path starting in the top right hand corner, to the bottom left LRA (green polygon). The true robot position is shown as a translucent cyan, while EKF estimate is translucent dark green. A white cell implies nothing of interest exists, while black indicates a near certainty of an area of interest. The localization landmarks are denoted as black circles.

The effect of environmental complexity on theoretical guarantees

In order to analyze the effect of environmental complexity (existence and density of obstacles), a minimum of 30 trials were run on each of the three obstacle scenarios (maps). The robot begins exploration at the bottom center of the map, and attempts to explore the entire space. The robot is required to terminate inside a randomly chosen LRA with $\alpha \geq .95$. Landmark density was the same in each scenario, and relatively dense given the constrained laboratory environment. Attaining the LRA is considered a success, while encountering an obstacle, filter inconsistency, and an inability to attain the LRA are considered failures. The robot uses a rudimentary obstacle avoidance procedure: to avoid collisions, operation is ceased and a path is replanned if an obstacle is detected in the immediate path. Such obstacle detections are considered failures because the robot can attain the LRA by luck after many obstacle detections and subsequent re-planning stages.

The robot only replans if: 1) it reaches the LRA 2) it encounters an obstacle (due to an inaccurate pose estimate) 3) its EKF estimate becomes inconsistent and it fails to re-localize 4) it completes its planned path, but does not reach the LRA. Each re-planning stage utilizes an updated posterior estimate of the exploration space based on measurements taken by the SICK.

Figure (2.9) shows the results of these experiments. The robot's Desired Achievement is

shown by the solid green line. The blue bars in Fig. (2.9) show Predicted Achievement of just over .98 in each case. The scenarios' setup and the result in Fig. (2.9), imply that paths of similar high Predicted Achievement were found between scenarios. The Realized Achievement from 30 runs is shown in yellow.

In the no obstacle case, three failures were observed where one failure was due to the robot exiting the defined map. This failure can be seen as an obstacle collision (the obstacle here being the map boundary). Controlling for this event, the realized achievement is $\approx .93$. Comparing this result with the simulation results in Fig. (2.6) provides valuable insight. In the simulation environment, maps were generated randomly and were more complex, in obstacle number and area coverage, than any of the experimental setups. In addition, the simulated robot, while following the same motion and measurement models, had no obstacle avoidance procedure. Thus, just as in the experimental procedure, any obstacle collisions were deemed failures. Whereas the simulations were conservative in complex environments, the experiment shows that the effects of model mismatch and un-modeled sensor noise made the constraint imposed the G-PIE and RH-PIE slightly optimistic.

A dramatic drop off in performance is seen in the obstacle scenarios in Fig. 2.9. The analysis of the no obstacle case and experimental data imply that this drop off in performance is due to poor intermediate localization which triggered the obstacle avoidance procedure. Thus, as a practical point, the performance of the G-PIE and RH-PIE is not agnostic to the particular implementation of obstacle avoidance. In addition, this result shows that, in contrast to claims made in [20, 31], path completion is not always dominated by terminal covariance. This dominance assumption can be fragile when sparse localization measurements are available for a non-linear system.

The fact that path completion can be dominated by intermittent positional error is a valuable finding. Again, in simulation, the RH-PIE and G-PIE algorithms are able to attain conservatism

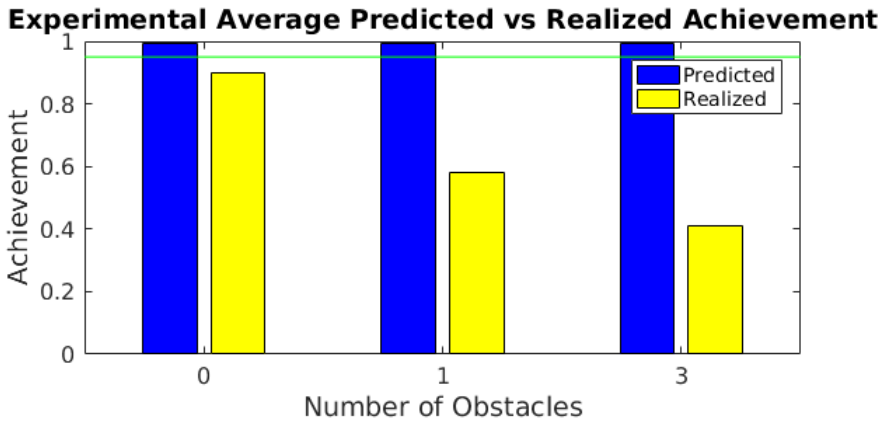
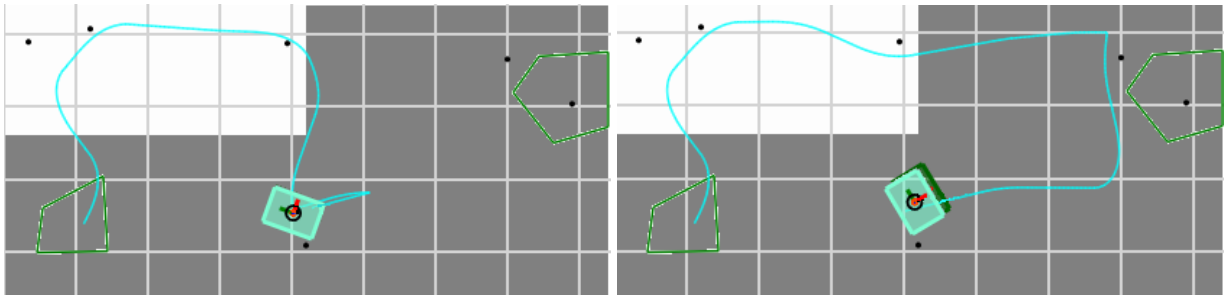


Figure 2.9: This figure shows the difference between Desired and Realized Achievement for three different obstacle scenarios.



(a) $T_1 = 1$ (b) $T_1 = 3$

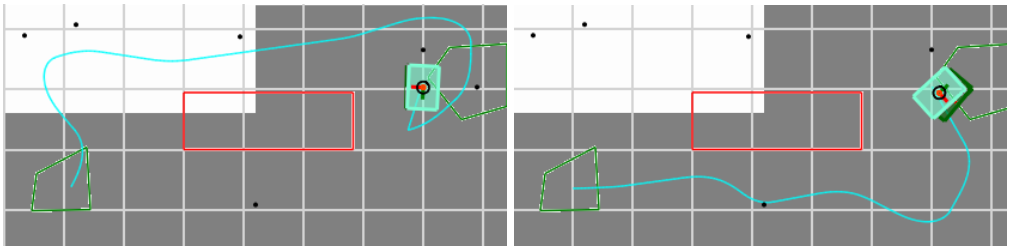
Figure 2.10: Two cases of the same obstacle scenario with the same initial conditions (IC1), but differing time horizons T_1. The robot is attempting to traverse the cyan path starting in the bottom center, and planning to the bottom left Localization Rich Area. $\beta = 0.1$.

in more complex scenarios, as shown in Fig. (2.6). This finding is consistent with simulation results provided in [20, 31]. Conversely, the experimental data suggest that metrics based on path completion, such as those presented here and in [20, 31], are not sufficient to ensure conservatism in real world scenarios. In particular, such path completion metrics should be augmented with direct consideration of intermediate localization. In addition, path completion metrics' interactions with obstacle avoidance procedures should be studied if they are to be practically useful and provide conservative results.

The effect of look ahead distance on path reward

To analyze the effect of look ahead distance on performance a set of experiments and a Monte-Carlo study were conducted. In the experiments, the robot starts in three different initial conditions in the same three exemplary maps, and $\beta = 0.1$ is constant; thus any change in expected entropy reduction is solely due to the change in T_1. The look ahead distance varied from 1 to 3 nodes away from the starting location (0.5-3m). Each large grid cell in Fig. 2.10 is $1\mathrm{m}^2$. Each data point represents the average of 3 trials: a total of 81 trials were performed. In this case any variability within a trial set is due to inevitable variation in the exact initial position.

Figure (2.12) plots expected entropy reduction as a function of T_1. As expected, an increase in the look ahead distance of the RH-PIE algorithm results in increased expected entropy reduction monotonically in all cases. In each map and set of ICs, there are two rates of increase: slow increases as seen in the single obstacle case, and more dramatic increases as seen in the obstacle free and three obstacle case. To understand these rates more clearly, consider Fig. (2.10), which shows the same scenario for two different horizon lengths. With a single step look ahead policy, the robot fails to take into consideration the localization landmark at the top right hand corner and subsequently takes a less exploratory path. With a two step look ahead policy, the robot sees this landmark and dramatically changes the way in which it navigates back to the LRA. This helps the robot to achieve a much longer and more information rich path. Conversely, in the case of IC3 in Fig. (2.12a), the robot simply takes one more exploratory step along $\mathcal{G}$, but does not fundamentally change its tail strategy. Figure (2.12) also implies that, for a high density of localization landmarks, such large differences in expected entropy reduction would not be seen between increments of the horizon T_1. This is intuitive because, with highly dense landmarks, regional values of B_{pos} would be similar, and the robot would not favor one area of the map over another due to localization. Finally, to analyze the difference between G-PIE and RH-PIE planned trajectories, a Monte-Carlo study using 115 randomly generated 5m$\times$5m maps with 2 polygonal obstacles and a PRM with 25 modes was conducted. Results show that, the average



(a) $T_1 = 1$ (b) $T_1 = 3$

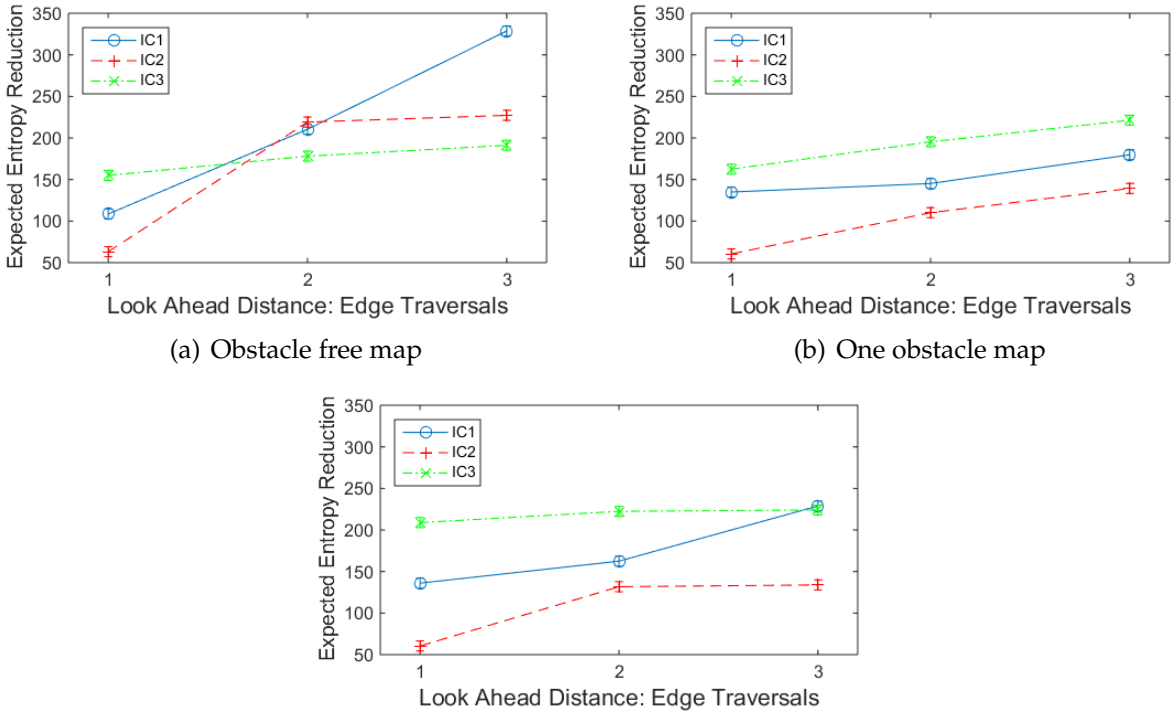
Figure 2.11: Two cases of the same obstacle scenario with the same initial conditions (IC3), but differing time horizons T_1. $\beta = 0.1$.

optimal G-PIE entropy reduction was 295.3 nats and the average optimal path length in node visitations was 12.7. The relative sub-optimality of the RH-PIE for $T_1 = \{3, 4, 5\}$ was 77.5%, 83.4%, and 88.3% respectively.

2.5.4 Relationship between environmental complexity and look ahead distance

Finally, consider the effect of obstacles on the RH-PIE and G-PIE. Figure (2.11) exemplifies this relationship.In Fig. (2.11), by increasing the look ahead distance T_1, the robot is able to find a more information rich path in a different homotopy class; below the obstacle instead of above the obstacle.

The change in path occurs because shortest path computation of R_{tail} can begin from nodes which are within sensor range of the bottom central landmark. Even though the path in Fig. (2.11a) is part of the subset of paths found for $T_1 = 3$, the path in Fig. (2.11b) is more information rich while still being feasible. This dramatic change, is similar to that seen by varying β, with the exception that the tail path remains localization seeking at $\beta = 0.1$. Thus, no looping behavior like that seen in Fig. (2.8b) occurs. This is despite the fact that the loop in the bottom center shown in Fig. (2.8b) exists in Fig. (2.11b).



(a) Obstacle free map

(b) One obstacle map

(c) Three obstacle map

Figure 2.12: This set of figures shows the growth in expected information gain as a function of increasing look ahead distance. The parameter $\beta = 0.1$ is constant. The maximum error due to IC variation is denoted by error bars.

Computation

Finally, it is important to note the near-real-time performance of the RH-PIE algorithm. All code was written in C# on a Windows 7 operating system. All code was run on a mobile I5 Intel Sandy Bridge processor. For $T_1 = 1$, the RH-PIE algorithm is able to calculate appropriate edge weights for $\mathcal{G}$ and plan a path in $\approx$ 30sec for a graph with 80 nodes. This is done while the robot is also processing sensor data, performing obstacle avoidance, and running visualization software. In contrast, the full G-PIE algorithm required $\approx$ 20min on the same hardware and a graph of 25 nodes, while not processing any auxiliary data. With a look ahead distance of 2, the RH-PIE algorithm is provides a solution in $\approx$ 1.5min while a distance of 3 requires $\approx$ 5min. This timing data implies that code optimization and a GPU implementation of the RH-PIE algorithm will allow real time replanning.

Summary and Discussion

The RH-PIE algorithm performs in an intuitive manner, trading localization and exploration as a function of β, as shown in Fig. (2.8). The optimality of the solution is traded in favor of practical speed as a function of the look ahead distance, exemplified by Fig. (2.10).

The tail reward function presented here provides a beneficial trade between conflicting objectives, constraining their values to guarantee tail optimality (in terms of shortest paths over $\mathcal{G}$), as in Fig. (2.8a). By loosening this restriction, the RH-PIE algorithm is able to return longer paths which more fully explore the space, as shown in Fig. (2.7, 2.8). In addition, careful implementation of shortest path algorithms can guarantee that the path, X_{best}, is returned in finite (polynomial) time. Thus, even though the tail $X_{T_1:T}$ is suboptimal, loosening the constraints on β provides informative paths while still guaranteeing the robot can re-localize upon path termination.

While locally optimal, the RH-PIE algorithm cannot make theoretical guarantees on global optimality due to the non-additive, path dependent nature of information. The RH-PIE still enables near real time performance with short look ahead distances. In addition, its simplicity and parallelizable structure allows for the majority of computation time (over 80%) to be simple matrix manipulation. Thus, real time re-planning can currently be achieved by leveraging parallelism and GPU acceleration.

The results in Fig. (2.9) provide evidence that the assumption that successful path completion is dominated by terminal covariance, as claimed in [20], is not well studied. Intermediate uncertainty in robot position, obstacle collision probability, and the interaction between obstacle avoidance and algorithmic guarantees are paramount in achieving practical conservatism in both the G-PIE and RH-PIE algorithms. The graph $\mathcal{G}$ in both algorithms could be extended to ensure obstacle avoidance using chance constraints similar to that presented in [13].

2.6 Conclusions

An information exploration planner, the Guaranteed Probabilistic Information Explorer (G-PIE) has been presented. The G-PIE algorithm solves the Integrated Exploration (IE) problem with probabilistic guarantees of path completion and asymptotically optimal exploration. An information based reward function is developed using entropy, which provides the flexibility to include a variety of exploration objectives. A formal bound to the information reward function is also developed for partially known environments. This bound enables fast computation of path rewards, reducing the computation time of the general problem by a factor of 10^3. A novel connection is made between the Hamiltonian Path problem and general exploration tasks which restrict allowable paths to a graph. Thus, all non-exhaustive exploration planners, such as belief planners, on general graphs $\mathcal{G}$ cannot provide any guarantee on exploration performance. Simulation results show that the G-PIE behaves in an intuitive manner, exploring the unknown area while fulfilling the required terminal localization constraint conservatively.

A computationally tractable, locally optimal approximation algorithm (RH-PIE) is also developed. The RH-PIE algorithm uses a receding horizon approach to give a locally optimal, information rich path. The RH-PIE algorithm guarantees that any returned path satisfies a constraint on re-localization. In addition, the RH-PIE algorithm provides a polynomial time approximation to the NP-hard longest path problem for robotic information gathering by utilizing a tail reward approximation which balances robot localization and information gathering. This balance is crucial in maintaining low pose uncertainty throughout a path and thus helps ensure the proper operation of low level controllers which rely heavily on accurate state estimates.

Real world experiments demonstrate that the RH-PIE algorithm is able to generate paths which are both informative and ensure re-localization in controlled environments. The RH-PIE tuning parameter, β, is able to effectively trade between exploration and localization while keeping computation low. At the same time, experiments imply that the assumption that path com-

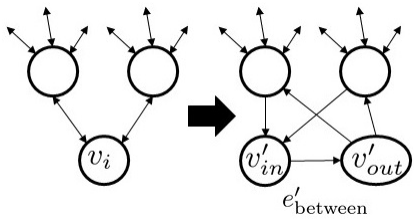


Figure 2.13: Exemplification of the difference between $\mathcal{G}$ and $\mathcal{G}'$

pletion is dominated by terminal covariance is incorrect. This finding reinforces the hypothesis that any objective function must balance information gain and localization to be of practical use.

2.A Proof of Thm. 2.3.5

Proof: Consider the Hamiltonian Path problem on $\mathcal{G}(V, E)$. Consider the modified graph $\mathcal{G}'(V', E')$ which is identical to $\mathcal{G}$ with the exception of an arbitrary node $v_i \in V$, which is replaced with v'_{out}, v'_{in}, where v'_{out} has all outgoing edges of v_i and only one incoming edge $e'_{\text{between}} = \{v'_{\text{in}}, v'_{\text{out}}\}$. Similarly, v'_{in} has only the incoming edges of v_i and the outgoing edge e'_{between}. The decision question is:"Does the longest path between v'_{out} and v'_{in} contain all verities in V'?". Clearly, $\mathcal{G}$ has a Hamiltonian path *iff* the answer is "YES". Now suppose there exists a polynomial time algorithm to approximate the maximization in Eq. 1 and 3 over $\mathcal{G}'$ within an error $\mathrm{e}_r \in \mathbb{R}$. Then, the PTA algorithm can solve the Hamiltonian Path problem on $\mathcal{G}$ [22]. But since $\mathcal{G}$ was arbitrary, the existence of such an algorithm implies that $\mathbf{P} = \mathbf{NP}$ ■

2.B Derivation of Prop. 2.3.6

Because of the cell independence assumption we just need an upper bound on $\mathbb{E}_Z[H(c|Z_1, ..., Z_n)]$ as a function of the number of sample points n. To reduce notation, define $\tilde{Z} \equiv \{Z_1, ..., Z_n\}$

By careful manipulation of this expectation using the definition of entropy and Bayes rule,

the following equivalence is evident:

$$\mathbb{E}_Z[H(c|\tilde{Z})] = H(c) + \sum_{c\in\{0,1\}} \Big[p(c)H(\tilde{Z}|c)\Big] - H(\tilde{Z}) \tag{17}$$

Therefore, in order to upper bound this quantity, a lower bound on $H(\tilde{Z})$ is sufficient. For large n, the distribution on $(\tilde{Z}|c)$ is approximately normal, thus by the Moivre Laplace theorem:

$$H(\tilde{Z}|c) \approx \frac{1}{2}\log\left(2 * \pi * e * n * \theta(1-\theta)\right) + \mathcal{O}(1/n) \tag{18}$$

The assumption is made that the entropy of a Gaussian mixture approximating a mixture of binomials converges to the entropy of a mixture of binomials. Thus, the third term can be approximated using the entropy bound in [32] as:

$$H(\tilde{Z}) \geq -\sum_{c} p(c)\log\left(\sum_{c'} p(c')\mathcal{N}\Big(\mu', \mu, 2n(\theta(1-\theta)\Big)\right) \tag{19}$$

where $\mathcal{N}(x, y, \sigma)$ denotes a normal distribution evaluated at y with mean x and standard deviation σ. Here, $\mu = n\theta$ if $c = 1$ and $\mu = n(1-\theta)$ otherwise. The same is true for μ' in terms of c'. Denote $\Delta\mu = \mu_1 - \mu_0$, where $\mu_0 = n(1-\theta)$ and $\mu_1 = n\theta$. Notice that the first term inside the log is simply the peak of a Normal distribution, thus the argument of the logarithm is:

$$\sum_{c'\in\{0,1\}} p(c')\mathcal{N}\Big(\mu, \mu', 2n(\theta(1-\theta)\Big) = \frac{p(c)}{\sqrt{4\pi n(\theta(1-\theta)}} + (1-\theta)\mathcal{N}(\Delta\mu, 2n(\theta(1-\theta)) \tag{20}$$

and

$$\mathcal{N}(\Delta\mu, 2n(\theta(1-\theta)) = \frac{1}{\sqrt{4\pi n(\theta(1-\theta)}} exp(-\frac{(2*n\theta - n)^2}{n\theta(1-\theta)}) \tag{21}$$

By using the properties of logarithms and analyzing the asymptotic properties of the member functions, Eq. (19) implies that for large enough n:

$$H(\tilde{Z}) \geq H(c) - \log\left(\frac{1}{\sqrt{K * n}}\right) \tag{22}$$

where $K = 4\pi\theta(1-\theta)$ is a constant w.r.t n.

Thus, by combining Eqns. (17), (18), (22) and passing to the limit, the result follows:

$$\lim_{n\to\infty} H(c) + \sum_{c\in\{0,1\}} [p(c)H(Z|c)] - H(Z) \geq \lim_{n\to\infty} \Bigg[H(c) + \frac{1}{2}\log\Big(2 * \pi * e * n * \theta(1-\theta)\Big) + \mathcal{O}(1/n) - H(c) + \frac{1}{2}\log\left(\frac{1}{Kn}\right)\Bigg] = \frac{1}{2}\log(\frac{e}{2}) \tag{23}$$

Thus the proof is complete if it can be shown that the entropy of a mixture of binomials approaches the entropy of a mixture of Gaussians. The proof of this is left out for brevity, but depends on discretion of the θ domain and careful application of the polynomial approximation theorem. ■

2.C Proof of the Eigenvalue Bound

In [20], the authors show that in an EKF setting, the aggregate update equations along a robot's nominal path are:

$$\Sigma_{k+1} = L + G(\Sigma_k^{-1} + H^T Q^{-1} H)^{-1} G^T$$

which resemble the standard Kalman filter equations. In the case where only one propagation/measurement pair is taken along an edge e_k, this equation reduces to the standard Kalman equations.

In the [33], Allen Knutson and Terrance Tao describe properties of a sum of Hermitian matrices. Suppose $A + B = C$. One property states that if λ_i, μ_i and ν_i are the ith eigenvalues of A, B and C respectively then:

$$\nu_{i+j+1} \leq \lambda_{i+1} + \mu_{j+1}$$

More specifically: $\nu_1 \leq \lambda_1 + \mu_1$.

Because Q and Σ_k are assumed to be positive definite Hermitian:

$$(\Sigma_k^{-1} + H^T Q^{-1} H)^{-1} \preccurlyeq \Sigma_k, (H^T Q^{-1} H)^{-1}$$

By these two observations, Eq. (12) is bounded by:

$$\lambda_1(\Sigma_{k+1}) \leq \lambda_1(L) + \min\{\lambda_1(G(\Sigma_k^{-1})G^T), \lambda_1(G(H^T Q^{-1} H)^{-1})G^T\} \tag{24}$$

where $\lambda_1(\mathbf{.})$ is the largest eigenvalue of its argument. It is important to note that this bound becomes tight as Q becomes small. This makes intuitive sense because Q small implies a near perfect observation. Notice that this bound relies on the invertability of $(H^T Q^{-1} H)$, or equivalently the full rank of H. This is an assumption on the observability of the robotic system along edge e_k. In general, this is not true in sparse landmark environments.

2.D Discussion of Information Approximation

This appendix derives two more methods for providing and information gain approximation as first discussed in Section 2.4.3.

A second approximation method partitions the exploration space into Voronoi regions about each edge e_i [24]. Once this is accomplished, any cell whose centroid is in the Voronoi region of e_i is associated with e_i. Let $M_{e_i}^{\text{under}}$ be the portion of the exploration space M associated with edge e_i through the Voroni regions. The tail information reward is:

$$B_{\text{info}}^{\text{under}}(X_{T_1:T}) = \sum_{e_i \in X_{T_1:T}} \Big[H(M_{e_i}^{\text{under}}) - \mathbb{E}_{Z_{e_i}}[H(M_{e_i}^{\text{under}})] \Big]. \tag{25}$$

This is a fast under-bound of the information gained from traversing an edge, but it may give large under estimates to a subset of edges.

A final approximation technique attempts to compensate for the weaknesses of the first two methods. In this approximation, the parts of the exploration space which are in range of several edges are penalized, but all cells in the sensor range of e_i are associated with e_i.

$$B_{\text{info}}^{\text{ave}}(X_{T_1:T}) = \sum_{e_i \in X_{T_1:T}} \left[H(M_{e_i}) - \mathbb{E}_{Z_{e_i}} \Big[\sum_{c_j \in M_{e_i}} c_j / k_j \Big] \right] \tag{26}$$

where k_j is the number of edges which have c_j in their sensor range. The value in Eq. (26) is neither an under nor over estimate of the information gained along e_i. To see this, consider the case when all edges within range of c_j are traversed by a single path. The concavity of the expected entropy causes an over estimate by Eq. (26). Conversely, if a path only traverses a single edge affecting c_j and $k_j > 1$, Eq. (26) produces an underestimate.

CHAPTER 3

JOINT EXPLORATION AND TRACKING: JET*

3.1 Introduction

The problems of exploration and tracking are tightly coupled in many real world scenarios, including surveillance, Search and Rescue (SAR), and defense. In some SAR tasks, robots first need to locate potential victims and then subsequently track them if they are moving; examples include victims in burning forests, in the ocean, or in an alpine avalanche. Furthermore, the tracking task - that of maintaining object locations - typically takes precedence over the exploration task. While exploration and tracking have typically been solved in separate stages to simplify the joint problem's complexity, this decoupled approach requires ad-hoc switching between stages which in turn makes guaranteeing tracking performance difficult. This work seeks to solve the joint problem of exploration and tracking under a unified framework, while guaranteeing tracking performance.

As background, research has been dedicated to the exploration problem with both single robotic agents as well as groups of homogeneous and heterogeneous agents; see [34] and the references within. Exploration applied to SAR includes 'probabilistic search', but does not usually consider tracking; several recent surveys have explored the state of this literature [34, 35]. This problem can also be framed as a mapping problem, as in [36]. Of particular relevance to this work, Al Khawaldah *et al.* consider the multi-robot exploration problem, but do not consider the detection or tracking of Objects of Interest (OIs) [37]. Similarly, Mottaghi and Vaughan use a particle filter to inform a team of robots of how to maximize the probability of detecting an OI [38], but also do not consider tracking.

A relevant variant of the exploration problem is termed the *coverage* problem, which implies ensuring that the largest possible area is sensed or 'covered'. Most work on coverage is inapplicable but, Pimenta *et al.* generate a continuous time algorithm which seeks to guarantee both coverage and tracking [39]. Although [39] attempts to solve these problems simultaneously, no guarantees of tracking performance are given, no information theoretic sense of exploration is

used, and the system is fully deterministic. Elston *et.al* also consider a multi-layered joint coverage and tracking problem [40]. Their work focuses on teams of 'mother ships' and 'daughter ships' which utilize a heuristic to partition a search area. Exploratory information is encoded by a heuristic, and robotic agents switch between pure tracking and pure coverage tasks.

A second related research area is the tracking problem from the viewpoint of traditional tracking metrics. Of particular interest, Ferrari *et al.* consider the tracking problem in a geometric manner and develop a closed form solution to the probability of detecting OIs under linearity and observability assumptions [41]. How *et al.* develop an RRT based planner which seeks to maximize track detections while achieving a goal [42]. Multi-agent multi-object problems have also considered questions such as data association and track assignment [34], but, in general, the tracking problem has been considered independent of the exploration or track detection problem.

Crucially, the literature rarely considers the joint exploration and planning problem. Instead, the joint problem is usually solved in two stages which are assumed independent (i.e. first explore then track). This two-stage approach has several drawbacks. First, tracking accuracy may not be well maintained in the search phase, and OI tracks may be lost when the tracking phase begins. In many problems, robotic agents do not definitively finish the search task and continually find more OIs over time. This then necessitates switching between phases in a potentially ad-hoc way. Finally, the two stage approach fails to exploit the full ability of robot agents because agents may have excess control with which to continue to search for more OIs (victims) while maintaining track accuracy of already located OIs.

In this work, the joint exploration and tracking (JET) problem is presented under a probabilistic framework which enables the problem to be solved in a single stage. This allows agents to utilize their full control authority, while maintaining tracking accuracy, and seamlessly transitioning between exploration and tracking. In addition, chance constraints are utilized to guar-

antee tracking performance, as in many applications such as SAR, tracking is paramount. An exploration objective function is derived which generalizes others found in the literature, e.g. [41]. Finally, a hierarchical formulation is presented which enables the continuous optimal control problem to scale well, be solved efficiently, and maintain guaranteed tracking performance. Simulation results show the efficacy of the JET approach.

3.2 Problem Formulation

The JET problem is posed as an optimization using an information metric and probabilistic constraints. Because robotic agents must first locate OIs, the performance metric, J_{info}, must incorporate detection up to a horizon time T. In applications such as SAR, tracking of *all* detected OIs must be maintained, and is therefore posed as a constraint.

The system adheres to dynamic equations:

$$\begin{aligned} \dot{x}_i(t) &= f_i(x_i(t), u_i(t), \omega_i(t)) \qquad i \in \{1, .., n\} \\ \dot{a}_j(t) &= F_j a_j(t) + \omega_j(t) \qquad\quad j \in \mathcal{T} \bigcup \hat{\mathcal{T}} \\ z_{i,j}(t) &= h_{i,j}(x_i(t), a_j(t), \nu_{i,j}(t)) \quad u_i(t) \in \mathcal{U}_i \end{aligned} \tag{1}$$

where $f_i(\boldsymbol{.})$ and $F_j(\boldsymbol{.})$ are the dynamics of robotic agents and OIs respectively. Note that LTI dynamics and Gaussian noise are assumed only for OIs. The sets $\mathcal{T}$ and $\hat{\mathcal{T}}$ denote tracked and untracked OIs respectively. It is assumed that each robot can localize itself via a nonlinear Kalman Filter (KF), and each robot's state estimate, $\bar{x}_i(t)$, and covariance are available [43, 44]. The precise type of KF is inconsequential to the results in this work. Discrete-time measurements are utilized. Thus, $z_{i,j}$ returns a value only at discrete-time instances. Note: $t^k = k\Delta t$.

3.2.1 Objective Function

Because the framework of the problem requires agents to detect un-tracked OIs, an analysis of detection probability is in order. The probability of a single OI being detected at time k is given as:

$$P(\mathcal{O}_j^k = 1 | \boldsymbol{z}_j^{1:k-1}, \boldsymbol{u}^{1:k-1}) \tag{2}$$

where $(\mathcal{O}^k = 1)$ denotes a positive detection at time instant k, $\boldsymbol{z}_j^{1:k-1}$ is a vector of all sensor measurements taken of the particular un-tracked OI up until the current time instance $k-1$, and $\boldsymbol{u}^{1:k-1}$ is the sequence of controls given to the robotic agents up until time $k-1$ (a zero-order-hold assumption). The **bold** notation implies that this is an aggregated vector of all similar variables, i.e. $\boldsymbol{u}$ is an aggregate of the controls of all robotic agents. This expression can be decomposed using the law of total probability un-marginalize the robotic and un-tracked object states. Then, by sequentially conditioning the detection likelihood, robotic state, and object state on all other variables, Eq. (2) can be shown equivalent to:

$$\int_{\boldsymbol{x}^k \in \boldsymbol{\mathcal{X}}^k} \int_{a_j^k \in \mathcal{A}_j^k} \Big(p(\mathcal{O}_j^k = 1 | \boldsymbol{x}^k, a_j^k) \times p(\boldsymbol{x}^k | \boldsymbol{u}^{1:k-1}) p(a_j^k | \boldsymbol{z}_j^{1:k-1}, \boldsymbol{u}^{1:k-1}) \Big) d\boldsymbol{x}^k da_j^k \tag{3}$$

where, $\boldsymbol{x}^k$ is the state of the robotic agents at time k, and a_j^k is the state of the jth OI. Notice that the first term of (3) is the detector model while the second and third terms are the predictive agent and OI distributions respectively. The full derivation of (3) is shown in Appendix 3.A.

To understand the detection probability - Eq. (3)- more easily, consider that there is only one agent and its position is perfectly known (i.e. the integral with respect to x^k disappears). In addition, suppose there exists a perfect OI detector with a circular field of view of radius r. In this case, Eq. (3) reduces to:

$$\int_{(a_j^k | d(a_j^k, x^k) \leq r)} p(a_j^k | z_j^{1:k-1}, u^{1:k-1}) da_j^k \tag{4}$$

In this case, the probability of detection simplifies to be the probability that the OI is within sensor range of the agent. The detection probability in (4) corresponds to, and therefore (3) generalizes, the sensor function in [41].

Using (3) as an objective function maximizes the probability that a single OI is detected at the next time instant k. Let $\boldsymbol{x}(t)$ and $\boldsymbol{a}(t)$ be the states of all robotic agents and OIs at time t. The variable $\boldsymbol{z}_j^{1:k-1}$ is a vector of sensor measurements taken of object j up to time t^{k-1}, and $\boldsymbol{u}(t)$ are the controls given to the robotic agents. The variables $\boldsymbol{\omega}(t)$ and $\boldsymbol{\nu}(t)$ are stochastic noise affecting the state of the mobile objects (robots and OIs), and measurements respectively. In this scenario, J_{info} is written as:

$$\begin{aligned} &J_{\text{info}}(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{z}, \boldsymbol{u}, \boldsymbol{\omega}, \boldsymbol{\nu}, k) := \\ &\quad \sum_{j \in \hat{\mathcal{T}}} P(\mathcal{O}_j^{k:K} = 1 | \boldsymbol{z}_j^{1:k-1}, \boldsymbol{u}^{1:k-1}) \end{aligned} \tag{5}$$

Note that the dependence of the arguments on time is suppressed for compactness and denoted by the argument k. The summation is taken over the set of un-tracked OIs $\bar{\mathcal{T}}$. The total number of tracked and un-tracked OIs, m, is unknown, finite, and assumed to be $m \leq n$. The challenge that m is unknown is addressed when discussing the hierarchical approximation of this problem in section 3.3.2.

3.2.2 Constraint on Tracking Performance

In traditional tracking, objects are typically tracked using a selection from a set of standard tracking estimators, referred to as Kalman Filtering (KF) techniques, which includes variants for both linear and non-linear dynamics [43]. In this work, tracked OI states are estimated using a KF,

which models the state transition and measurement likelihoods as Gaussian distributions. As such, tracking performance can be analyzed via the covariance matrix:

$$\Sigma_j^k = \mathbb{E}[(\bar{a}_j^k - a_j^k)'(\bar{a}_j^k - a_j^k)] \quad \forall j \in \mathcal{T} \tag{6}$$

where $\bar{a}_j^k$ is the estimate of the jth tracked OI at k.

To guarantee tracking of discovered OIs, a bound on the covariance, Σ_j^k, must be satisfied for every tracked OI. Since all real-world sensors are imperfect, there is a chance that the sensor does not detect/measure a tracked OI at a particular time instance, i.e. missed or intermittent detection. The presence of intermittent measurements implies that no deterministic bound can be given for Σ_j^k; instead, a bound on the expected value is used, i.e. $\mathbb{E}\big[\Sigma_j^k\big] := \int p(\Sigma_j^k|\mathbf{Z}_j^{1:k})p(\mathbf{Z}_j^{1:k})dZ$.

For cases of linear dynamics and intermittent measurements, the Algebraic Riccatti Equation is a contraction and can be used to guarantee the existence of $\mathbb{E}\big[\Sigma_j^k\big]$ [45], [46] ,[47]. Through these results, it can be shown that, for a certain range of probabilities of detection $P(\mathcal{O}_j^k = 1)$, the tracking error covariance $\mathbb{E}\big[\Sigma_j^k\big]$ is bounded uniformly in t and there exists a finite steady state distribution for Σ_j^k [45, 47]. Thus, one only needs to bound the probability of detecting an OI in order to provide a bound on its expected covariance matrix.

Consider now the following bound on the tracking error of tracked OIs,

$$P(\mathcal{O}_j^k = 1|\boldsymbol{z}_j^{1:k-1}, \boldsymbol{u}^{1:k-1}) \geq 1 - \alpha \quad \forall j \in \mathcal{T} \tag{7}$$

where $\alpha \in (0, 1)$ is a user set parameter. Notice that this bound utilizes the same expression as the objective function in Eq. (2). The constraint in (7) simply says that the joint robotic system has at least a probability of $1-\alpha$ of seeing OI j up to the discrete time instant k, given the measurement and control histories. Assuming that OIs follow LTI dynamics with additive Gaussian noise, the

results in [45, 47] can be leveraged and $\mathbb{E}\left[\Sigma_{a_j^k}^k\right]$ can be computed for the chosen α. This calculation can be done via direct Monte-Carlo simulation is in the results here, or by using the bound derived in [45]. The proposed optimal control problem is then: maximize (5) subject to (1,7). This formulation implies that robots are opportunistic explorers, but must maintain tracking.

3.3 A Hierarchical Approximation

The problem presented in Section II is computationally heavy and impractical for real time control. In addition, the problem does not fulfill the standard DP assumption of an additive, reward J_{info} . Finally, the dimensionality of the control and state spaces increases linearly with number of robotic agents and OIs, and trajectory optimization scales poorly with state dimension [48] [49].

To make the JET problem real-time tractable, a hierarchical approximation is presented. The hierarchical framework first provides optimal Next-Best-View (NBV) positional goals at horizon time T to each robotic agent. A lower level nonlinear optimization then solves the continuous time optimal control problem for each agent independently and satisfies the dynamics. This approach sacrifices information optimality by only coordinating the terminal location of robots, but still solves the exploration and tracking problem jointly and provides probabilistic tracking guarantees.

3.3.1 Reduction to the NBV problem

The NBV optimization is designed to give fast approximate exploration goals, at the time horizon T, to each robotic agent. These goals seek to maintain tracking performance. Three assumptions reduce the problem in Section II to the NBV problem. First, note the objective in Eq. (5) is depen-

dent on the trajectory history $(\boldsymbol{x}_{0:T}, \boldsymbol{a}_{0:T}, \boldsymbol{z}_{0:T})$. This motivates the use of an Open-Loop-Feedback (OLF) strategy in which expected intermediary measurements are ignored [48], therefore we assume no measurements are taken between the initial time t^0 and the terminal time T. Second, to make (5) additive, we assume no 'information overlap' occurs between robotic agents when the agents are 'well spaced'; i.e. if $||x_i^K - x_j^K||$ is large enough then $P(\mathcal{O}_j^K = 1|\boldsymbol{z}_j^{1:k-1}, \boldsymbol{u}^{1:k-1}) \approx \sum_{i=1}^{n} P(\mathcal{O}_{i,j}^K = 1|\boldsymbol{z}_j^{1:k-1}, \boldsymbol{u}_i^{1:k-1})$. Last, a coarse, discrete, linear approximation of the robotic dynamics is assumed. Formally: $\exists E \subset \mathbb{R}^{n_x}, \mathrm{U} \subset \mathbb{R}^{n_u}, B \in \mathbb{R}^{n_x \times n_u} \quad s.t. \quad E = \{x : x = B\mathrm{u}, \mathrm{u} \in \mathrm{U}\}$, and $\forall x \in E$, x is reachable by the nonlinear system from the origin in time T. This is a local controllability assumption. Work has been done on approximating local reachability [50]. In this work, an optimal linear approximation is not derived, but, for the unicycle model used here, a simple analysis can yield a coarse approximation readily [44]. In the sequel, note that $K\Delta t = T$. Finally, to help satisfy constraint (7) at K, linear observability at K through $H_{i,j} \in \mathbb{R}^{n_x \times n_z}$ and a linearized transition $F_i \in \mathbb{R}^{n_x \times n_x}$ are assumed. Thus, the coarse dynamics and measurement predict motion up to the time horizon T:

$$\begin{aligned} x_i^K &\approx F_i x_i^0 + B_i \mathrm{u}_i^0 + \mathrm{w}_i^0 \\ a_j^K &= F_j^K \cdot a_j^0 + \mathrm{w}_j^0 \\ z_{i,j}^K &= H_{i,j}(x_i^K - a_j^K) \cdot (x_i^K - a_j^K) + \mathrm{v}_{i,j}^0 \\ \mathrm{u}_i^0 &\in \mathrm{U}_i \end{aligned} \tag{8}$$

The variables (w, v) are the time integrals of their continuous time noise counterparts [43]. Roman notation, (e.g. u vs u), denotes a discrete time counterpart of a variable. Because this coarse approximation follows approximate linear dynamics, the control sets $\mathcal{U}_i$ and U_i are not the same. Note that the OIs are assumed to follow LTI dynamics, which allows for the application of bounded expected covariance due to (7).

3.3.2 An approximate objective function for NBV goals

Given the coarse dynamics and measurement prediction in (8), the higher level exploration problem seeks to maximize:

$$\max_{\mathbf{u}^0 \in \mathbf{U}} \hat{J}_{\text{info}}(\boldsymbol{x}^0, \boldsymbol{a}^0, \boldsymbol{z}^0, \mathbf{u}^0, \mathbf{w}^0, \mathbf{v}^0, t^0) = \max_{\mathbf{u}^0 \in \mathbf{U}} \sum_{j \in \hat{\mathcal{T}}} P(\mathcal{O}_j^K = 1 | \boldsymbol{z}^0, \mathbf{u}^0) \tag{9}$$

Equation (9) maximizes detection of untracked OIs at the horizon time K. More specifically, the optimal control, $(\mathbf{u}^0)^*$, produced by maximizing Eq. (9), generates an optimal set of next-best viewpoints $(\boldsymbol{x}^K)^*$.

3.3.3 A tighter tracking constraint

The NBV formulation, while at a coarser level, must continue to guarantee tracking for each OI in $\mathcal{T}$. Instead of satisfying Eq. (7), the NVB formulation requires that a robotic agent is assigned to each OI which is actively being tracked. The assigned agent is then required to guarantee an observation of its OI at the horizon time T. This constraint is formally defined as:

$$\begin{aligned} &\exists i \in \mathcal{A}, \qquad \forall j \in \mathcal{T} \quad s.t. \\ &P(\mathcal{O}_{i,j}^K = 1 | \boldsymbol{z}_j^{1:k-1}, \mathrm{u}_i^{1:k-1}) \quad \geq \quad 1 - \alpha \end{aligned} \tag{10}$$

where $\mathcal{A}$ is the set of assigned robots. Equation (10) is a tighter constraint than Eq. (7), and the assignment of agents necessitates the assumption that $m \leq n$. The higher level NVB problem is summarized as: maximize Eq. (9) subject to Eqs. (8,10). The NVB result is a set of optimal Next-Best-View points $(\boldsymbol{x}^K)^* = \boldsymbol{x}^*(T)$. As $T \to \Delta t$, the NVB problem guarantees a probability of detecting known OIs at each time step, but greatly reduces the exploratory capability of agents, and makes exploration myopic. If robotic agents have overlapping sensor fields of view at time T, the mutual information between agents must be considered. This problem is related

to Distributed Data Fusion and can be a difficult to solve [51]. Instead, via assumption two, an additional 'well spaced' constraint separates agents by M, a positive sensor-dependent constant, at the terminal time:

$$||\bar{x}_i^K - \bar{x}_j^K|| \geq M \quad \forall i \neq j \quad i,j \notin \mathcal{A} \tag{11}$$

3.3.4 Distributed optimization

Given the maximizer of the higher level problem, $(\boldsymbol{x}^K)^* = \boldsymbol{x}^*(T)$, which implicitly assigns the robotic agents to OIs, the low level problem is considered. The constraint in Eq. (10), along with the results in [45, 46, 47], ensures that tracking performance is maintained for a short time horizon T. Thus, the lower level problem no longer needs to consider tracking performance directly. Instead, the following constraint must be met:

$$\boldsymbol{x}(T) = \boldsymbol{x}^*(T) \tag{12}$$

The lower level optimization is then solved independently at each t^k, in a distributed fashion, by each robot. A single assumption, consistent with the OLF strategy, is needed to make the low level problem tractable. Recall that J_{info} is non-additive through time since there is 'information overlap' between a robot's sensor through time. This also means that J_{info} is time varying and also dependent on the previous path taken by robotic agents. The only way to accurately account for this time and state dependence would be to enlarge the state space of the optimization to include the time varying state of the cost function [48]. Although this is theoretically possible, the subsequent state space explosion makes state augmentation impractical. For the lower level problem, we instead take the line integral of the current probability of detection over the planned

robotic path. In other words:

$$J_{\mathrm{LL}} := \int_{x_i(t^k:T)} P(\mathcal{O}_j^k = 1 | \boldsymbol{z}_j^{1:k-1}, \boldsymbol{u}_i^{t^k:t}) dt \tag{13}$$

This is in accordance with assumption one in Section 3.3.1. Intuitively, Eq. (13) uses assumption one but positions the robot such that if a measurement is unexpectedly taken at any time $t \in (t^k, T)$, the robot will be in a locally optimal position. The lower-level optimization is now given by:

$$\begin{aligned}
\max_{u_i(t)} \quad & J_{\mathrm{LL}}(x_i, \mathbf{a}, z_i, u_i, \omega_i, \boldsymbol{\nu}, t^k) \\
\text{s.t.} \quad & \dot{x}_i(t) = f_i(x_i(t), u_i(t), \omega_i(t)) \\
& \dot{a}_j(t) = F_j a_j(t) + \omega_j(t) \\
& z_{i,j}(t) = h_{i,j}(x_i(t), a_j(t), \nu_{i,j}(t)) \\
& x_i(T) = x_i^*(T) \\
& u_i(t) \in \mathcal{U}_i, \quad t \in (t^k, T), \quad j \in \hat{\mathcal{T}} \cup \mathcal{T}
\end{aligned} \tag{14}$$

The lower level optimization is solved independently at each t^k, in a distributed fashion, by each robot. At each time step k, new information is incorporated in the posterior distribution of unknown object locations and the robots re-optimize (14).

3.3.5 The Joint Exploration and Tracking (JET) algorithm

A full description of a Joint Exploration and Tracking (JET) algorithm with probabilistic guarantees can now be given in Alg. (3).

The JET algorithm is structured intuitively. For compactness, all currently available information is denoted $\mathcal{I}(t)$, including estimates of robot states, OI estimates and distributions, and

1 $\mathbf{x}(0) =$ InitializeStates();
2 $p(\boldsymbol{a}) =$ InitalizeOiDist() ;
3 $\mathcal{T} = \emptyset$;
4 **while** *true* **do**
5 $\quad UpdateOiPdf(\mathcal{I}(t), \boldsymbol{z}(t))$;
6 $\quad$ **if** *New OI detected* **then**
7 $\quad\quad \mathcal{T} = \{\max(\mathcal{T}) + 1\} \cup \mathcal{T}$
8 $\quad$ **end**
9 $\quad$ **if** *(t%T)==0) or New OI detected* **then**
10 $\quad\quad$ **if** *(t%T)==0* **then**
11 $\quad\quad\quad t_0 = t;$
12 $\quad\quad$ **end**
13 $\quad\quad Assignment = SolveAssignment(\mathcal{I}(t))$;
14 $\quad\quad (\boldsymbol{x}^*(T + t_0)) = SolveNBV(\mathcal{I}(t));$
15 $\quad$ **end**
16 $\quad$ **for** $i \in \{1, ..., n\}$ **do**
17 $\quad\quad (x_i^*(t), u_i^*(t)) = SolvePath(\mathcal{I}(t))$;
18 $\quad\quad \mathrm{u}_i^* = u_i^*(t + \Delta t/2);$
19 $\quad$ **end**
20 $\quad t = t + \Delta t$;
21 **end**

Algorithm 3: Joint Exploration and Tracking: JET

previous controls. At each discrete time instant k, any new sensor measurements are used to update the OI distributions (line 5). If a new OI is detected (line 6), it is added to the detected set $\mathcal{T}$. Using the new detected set, the assignment problem is solved so that each detected OI is assigned a robotic agent, in order to satisfy Eq. (10) (line 13). In this study, the assignment is performed by solving the linear assignment problem using Euclidean distance from agents to expected OI locations at T as the cost. Different metrics for solving the assignment problem are possible and would result in switching behavior being manifest under different conditions [40].

Given the assignment, the high level, Next-Best-View, problem is then solved, Eqs. (8 - 11) (line 14), using current estimates of the OIs expected positions. The NVBs , $(\boldsymbol{x}^K)^*$, for each robotic agent are then used to solve the decentralized lower level path planning problem locally using direct transcription to account for the non-linear dynamics (line 17) [49]. Note that the high level NVB problem is solved centrally at each time horizon, or when new OIs are discovered; each

robotic agent can solve its own optimization independently in parallel.

3.4 Performance, Modeling, and JET Guarantees

Given the defined higher and lower level optimization problems outlined in Alg. (3), the specific modeling assumptions on the distributions of untracked and tracked OIs, robots' states, and the form of the detection function are required to enable full implementation.

First, consider the detection function in Eq. (3). Most current literature assumes a perfect detector, which is an indicator function within a detection range or region [41]. Real sensors are seldom perfect detectors, and their accuracy may drop off as a function of range. This decrease in accuracy is especially true when using sensors, such as LIDAR and cameras, which become less accurate or resolute at a further range.

Instead of an indicator function, this work models the idealized detector as an un-normalized Gaussian Mixture (GM), which has several advantages. First, the detection probability can be cast as a function of distance, which is appropriate for LIDAR and camera sensors. Second, GMs can be composed to model complex behaviors. Finally, GMs have continuous derivatives.

Formally, the state space of the OIs and agents are assumed the same, i.e. $n_a = n_x$, and the detector for tracked and un-tracked OIs is modeled as a quasi-concave GM where c_l are constant vectors:

$$P(\mathcal{O}^k = 1|x^k, a^k) = \sum_{l=1}^{n_s} \zeta_l \cdot \exp\left(-\frac{1}{2}(a^k - (x^k - c_l))' \Sigma_{O_l}^{-1} (a^k - (x^k - c_l))\right) \tag{15}$$

where:

$$\max(p(\mathcal{O}^k = 1|x^k, a^k = x^k)) = 1, \quad \zeta_l > 0$$

KF methods are used to estimate tracked OI and robot states yielding multivariate Gaussian

distributions.

$$\begin{aligned} p(x_i^k|u_i^{1:k-1}) &\sim \mathcal{N}\big(\bar{x}_i^k, \Sigma_i^k\big) \\ p(a_j^k|z^{1:k-1}, \boldsymbol{u}^{1:k-1}) &\sim \mathcal{N}\big(\bar{a}_j^k, \Sigma_j^k\big), \quad j \in \mathcal{T} \end{aligned} \tag{16}$$

Finally, versatility and smoothness make GMs a good candidate to represent the untracked OI distribution as well. Recent powerful tools which have been developed to approximate arbitrary distributions as GMs, [18], are of particular importance to the objective function in Eq. (5) because, if a robotic agent fails to detect an untracked OI at any time instant, a 'negative' measurement usually results in a posterior distribution which has no closed form. Therefore after every negative measurement, the resultant posterior distribution must be re-approximated as a GM using K-means clustering or another technique in order to maintain computational efficiency and avoid having to discretize the exploration space. As a result, unknown OIs are modeled using GMs:

$$p(a_{\mathrm{j}}^k|z^{1:k-1}, \boldsymbol{u}^{1:k-1}) = \sum_{l=1}^{n_d} \gamma_l \cdot \exp\left(-\frac{1}{2}(a_{jl}^k - \mu_{jl})' \Sigma_{jl}^{-1} (a_{jl}^k - \mu_{jl}) \right) \tag{17}$$

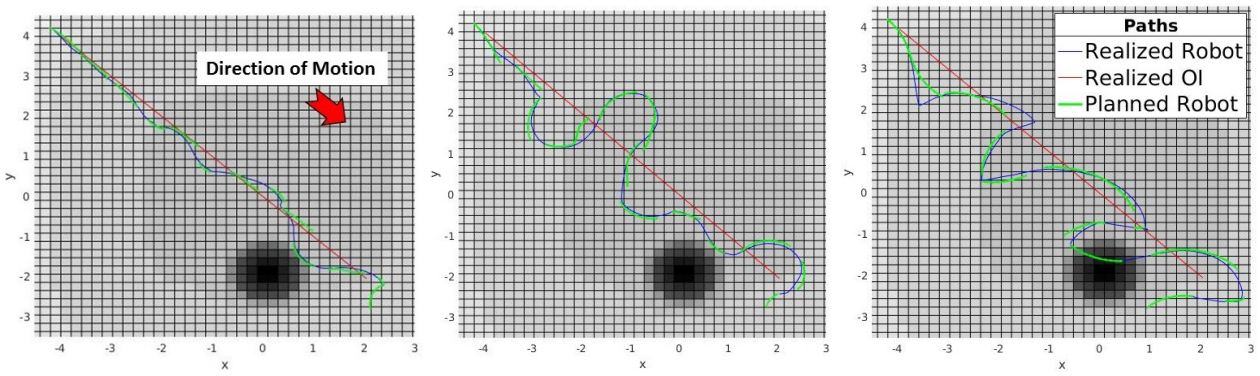
where:

$$j \in \hat{\mathcal{T}}, \quad \gamma_l > 0$$

Note that if tracked OIs are estimated using a Kalman filter, this implies that, as soon as a new OI is discovered and enters the tracked set $\mathcal{T}$, the tracked OI's distribution is no longer modeled as a the GM in Eq. (17). Therefore, *only* negative measurements are taken of the un-tracked OIs.

Given these modeling assumptions, solution characteristics emerge, which are instrumental in allowing the lower level path planning problem to be solved quickly using standard nonlinear optimization tools.

Proposition 3.4.1 *Given Eqs. (15, 16), if $n_s = 1$, then the feasible set for the assigned robot i in Eq.* (10) *contains a subset $\mathcal{S}$ which is convex in $\bar{x}_i(T)$ and quadratic in $||\bar{x}_i(T) - \bar{a}_j(T)||$*



(a) Excess Authority: 0.3m/sec (b) Excess Authority: 0.8m/sec (c) Excess Authority: 2.3m/sec

Figure 3.1: Robot behavior as a function of control authority.

The proof of Proposition (4.1) is found in Appendix 3.B. The convexity of the constraint provides a fast global feasibility check and is therefore valuable in making the JET algorithm real-time. For details on the difficulty of finding initial feasible points see Phase I methods in [52]. A similar, but weaker, result is true if the sensor is modeled by more than one mixand, assuming the detector GM is quasi-concave. This result allows for a much larger class of sensors to be accurately modeled while maintaining the convexity of the NVB problem. Both the proof and further discussion are in Appendix 3.B.

3.5 Simulation Results

To analyze the behavior of the JET algorithm, three simulation studies are conducted. The first study shows the behavior of the low-level optimizer as a function of the excess control authority given to a robotic agent. The second study analyzes the behavior of the high-level optimizer and its ability to distribute robotic agents and maintain tracking of OIs. The final study analyzes the behavior of the OI covariance matrix under the JET algorithm's assumptions.

3.5.1 Exploration as a function of control authority

In this case study, a single robot is tasked with maintaining track accuracy of a single OI which has already been detected. Furthermore, there is an *a priori* probability of another OI being in the space, which is nearly uniform except for a peak at the bottom right corner of the map. This study seeks to exemplify the behavior of the low-level planning component of the algorithm as additional control authority is made available. In addition, the study analyzes the effects of the Open Loop Feedback (OLF) approximation on the optimality of the resultant path.

Recall that an OLF strategy does not take into account any expected measurements of the environment in the future; as such, it is sub-optimal in general. Instead, with each new measurement of the map, the robot re-plans its path to incorporate the new information. Thus, the repeated re-planning addresses the sub-optimality of the OLF assumption. In this case study, the robotic agent moves deterministically and assumes process noise in the tracking model of the OI. The true behavior of the OI is deterministic to ensure repeatability of the experiment. In all cases, the following variables are kept constant: the prior distribution on untracked OIs, the realized tracked OI motion, and the initial conditions. The only variable is the control authority, $u_i(t)$, available to the robotic agent.

Figure 3.1 shows the results of this case study for three levels of control authority. The blue line is the robot's true realized path throughout the experiment, while the red line is the path taken by the tracked OI. The green lines in Fig. 3.1 show planned paths at particular instances in time separated by 0.5sec. Measurements of the environment are taken at $10Hz$ and the initial prior distribution on untracked OIs is overlayed as a heat map, with darker colors implying higher probability. Notice that, in each of the figures, the robot strays away from the expected OI path to explore the surrounding area. In all cases, the robot attempts to sense the *a priori* peak in the bottom right and stays to the left of the true object trajectory. As the control authority increases ($a \rightarrow c$), the robot is able to explore further from the expected OI trajectory, which

it must observe at intervals of 2.0sec. Subsequently, the planned green paths are longer. The planned and realized (blue) paths differ due to new measurements, which update the heat-map (not shown), but are very similar as measured in euclidean distance. This suggests that the OLF method approximates the optimal trajectory well even while ignoring future measurements.

3.5.2 Behavior and speed of the hierarchical approach

The second study seeks to qualitatively assess the ability of the JET algorithm to utilize a team of robots to both explore for unknown OIs and maintain localization of tracked OIs. The time horizon is set at $T = 2$sec and the detection/observation constraint probability for tracked OIs is set at $.65$ ($\alpha = .45$).

To avoid clutter and confusion, Fig. 3.2 shows a series of snapshots of planned agent paths at three successive times ($t \in \{0, 2, 8\}$). Each figure shows five robotic agents searching a 12×12 area for, and subsequently tracking, three unknown OIs. The OIs move with stochastic dynamics. Initially, no information is known about the number or location of OIs except that they are not near the current location of the robotic agents. As such, the same un-informative prior distribution is assumed for the location of each OI's initial location. Since un-tracked OIs are independent and have not yet been detected by definition, their distribution develops in *exactly the same way* due to the negative measurements taken by robotic agents. The recursively updated posterior distribution of all un-tracked OIs is represented as a GM distribution, as shown by the heat-maps in Figure 3.2, where warmer colors represent a greater likelihood of an un-tracked OI being located at that point.

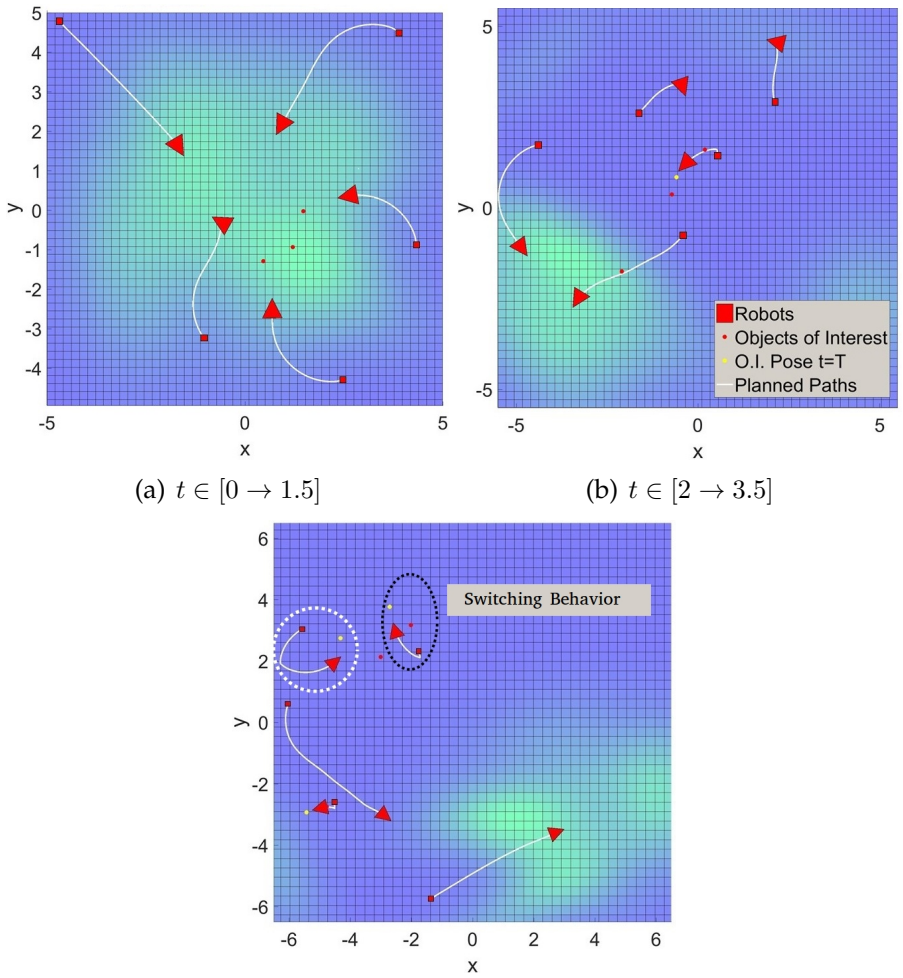
Agents are shown in red, with their initial position marked by a square and their planned final position and orientation shown as a triangle. True OI positions are shown as red dots, while estimated positions at T are shown as yellow dots. Figure 3.2(a) shows the initial condition

$(t = 0\text{sec})$ and all untracked OIs have no expected position. Figure 3.2(b) $(t = 2\text{sec})$ shows a single OI being tracked, while Fig. 3.2(c) $(t = 8\text{sec})$ shows all OIs being tracked.

The high level NBV optimization places the terminal positions of exploratory agents near the peaks of the unknown objects' GM distribution (light green) while keeping the robots' fields of view well separated. Conversely, at the time horizon T, assigned robots are required to be relatively near the expected terminal positions of their assigned OIs in order to satisfy the detection constraint. The low-level continuous time path planner then optimizes the robots' trajectories, thereby improving information gathering, while satisfying the robots' non-linear dynamics. Each figure shows that the terminal locations of purely exploratory robotic agents are well separated due to the separation constraint in Eq. (11). As a consequence, the planned exploratory trajectories tend to also be separated without requiring explicit coordination between robots at the low level.

A closer examination of the individual robotic paths (not shown) reveals a similar behavior to that seen in Fig. 3.1, where the robotic agent explores its environment when it has excess control authority, and tracks its assigned OI. The robotic agent's path crosses the OI's expected path approximately at each time horizon T. Figure 3.2(c) $(t = 8\text{sec})$ shows that after six time horizons, the robotic team successfully discovers all OIs and maintains tracking of each OI.

A new emergent 'switching' behavior from the JET algorithm can be seen in Fig. 3.2(c). A robot, near $(-2, 2)$, which is tracking an OI near the robot's current position, changes its role to track a newly discovered OI, near $(-2, 3)$. The robot and its new OI are encircled with dashed black lines. At the same time, a robot that was previously exploring is now tasked with tracking the 'old' OI (encircled with dashed white lines). Similar switching occurs when new OIs are discovered, and exploratory vs tracking roles are changed/updated.



(a) $t \in [0 \rightarrow 1.5]$ (b) $t \in [2 \rightarrow 3.5]$

(c) $t \in [8 \rightarrow 10.5]$

Figure 3.2: Simulation studying multi-agent behavior, with the transition from exploration to tracking. The predicted path is shown up to $T - .5$ to help visually distinguish expected OI positions

3.5.3 Distribution of tracked OI covariance

In [46], the author discusses the discrete behavior of the distribution of covariance matrices in LTI systems with intermittent measurements. The work here uses an LTI model of OI behavior, which is noise driven in the velocity states, and therefore meets the assumptions in [45, 46]. Figure 3.3 shows a histogram of 10,000 Monte-Carlo, covariance norms of a single tracked OI after 20 time horizons. The OI is intermittently measured at each time horizon. This is equivalent to assuming that the coarse-dynamics tracking constraint in (10) is met for all 20 time horizons.

In this study, a probabilities of detection of 0.65 and 0.75 are analyzed. In addition, the same

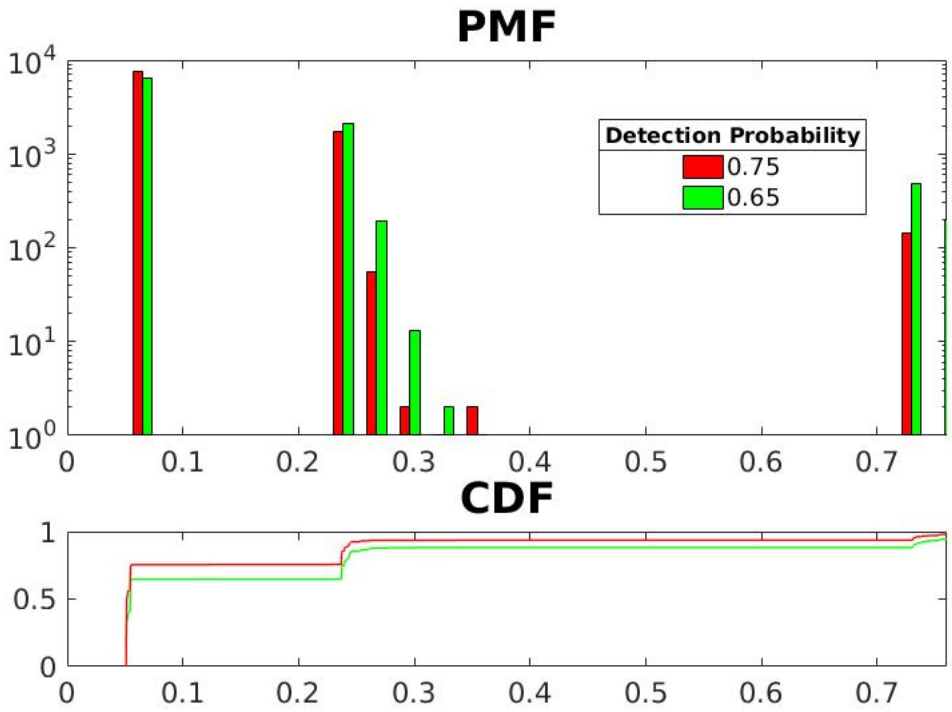


Figure 3.3: The empirical PMF and CDF of Σ norms at $t = 20T$.

time horizon of $T = 2$sec, and OI process and measurement noise statistics are used as compared to the prior studies. Figure 3.3 shows only the smallest 95% of samples to avoid poor figure scaling. Notice that peak of the PMF, representing 64.8% and 75.4% of all cases respectively, are clustered at $< 0.1\mathrm{m}^2$; this implies that the OI is still being well tracked. The means of the PMFs are $0.265\mathrm{m}^2$ ($1 - \alpha = .65$) and $0.154\mathrm{m}^2$ ($1 - \alpha = .75$). The CDFs show that 87.7% and 93.5% of covariance matrices respectively, have a norm less than $.27\mathrm{m}^2$. A norm of $.27\mathrm{m}^2$ indicates that the OI has a 99.7% chance of being within two sensor radii of its mean position. Summarizing all cases, if the probabilistic constraint of detection in Eq. (10) is met for each time horizon, a robotic agent maintains relatively accurate positional awareness of an OI in 87.7% and 93.5% respectively. This style of Monte-Carlo study can be used to inform a designer of the appropriate trade in time horizon length, tracking accuracy, and exploratory performance for a particular application of the JET algorithm.

3.6 Practical and Real Time Implications

The likelihood that OI tracking is lost if the constraint in Eq. (10) is met is small, yet the approximate hierarchical JET algorithm may lose tracking of an OI more frequently than the constraint implies. If the covariance of tracked OIs becomes large enough, a single robot may not be able to guarantee viewing its assigned OI with the required $(1-\alpha)$ probability. In this case, the problem becomes infeasible. If a tracked OI's covariance grows too large, a hybrid behavior is necessary to make the high level problem feasible once more. This hybrid behavior may require a tracking robot to switch to a pure tracking/following criterion and ignore the cost function in Eq. (14) completely until tracking accuracy is recovered. Fortunately, the convexity of the constraint in Eq. (10) allows for a fast feasibility check.

In addition, the lower level optimization is subject to limitations experienced by Non-Linear Program (NLP) solvers [49]. This work uses the MIT DRAKE toolbox along with a direct transcription method to solve the continuous time optimal control problem [53]. The initial guess provided to the NLP solver can make a large difference in computation time and even feasibility. In addition, the existence of a non-trivial gradient of the objective function in Eq. (5) is required for a timely solution. Although not common, a locally near-zero gradient may occur when a robotic agent is much faster than its assigned OI.

The average computational performance was evaluated on a 2.6Ghz Intel i7-6600U processor in MATLAB using the DRAKE toolbox and SNOPT as the underlying NLP solver. The combined time of high level problem with 5 agents and a single robotic agent's low level problem typically falls in a range of $(.5 - 6\text{sec})$. This excludes instances in which the NLP solver has difficulty converging due to a near-zero gradient, which can be compensated for by using appropriate problem scaling. In summary, these initial results show that the JET algorithm has potential as a real-time planner.

3.7 Conclusions

This work proposes a framework which enables the multi-robot multi-object problem to be solved simultaneously. The JET algorithm allows a team of robots to search for OIs while a probabilistic constraint on the tracked OIs' covariances guarantees tracking performance throughout the entire mission. Automatic discovery of new OIs, a seamless transition to guaranteed tracking of discovered OIs, and automatic balancing of exploration with the requirements of tracking are the primary novelties of the proposed algorithm. A novel hierarchical architecture is used to approximate the optimal control problem and coordinate robotic agents in the tracking of multiple OIs while simultaneously allowing the task to remain computationally efficient.

The JET algorithm enables each robotic agent to fully utilize its control authority to optimize exploratory behavior. At the same time, JET provides probabilistic guarantees on tracking performance, which are crucial in search and rescue scenarios. Simulation results show that the JET algorithm produces intuitive exploratory paths while maintaining tracking accuracy. In addition, the JET algorithm is able solve for continuous time optimal trajectories in a receding horizon fashion and has real-time performance potential.

This appendix provides derivations and proofs for some of the formulas discussed in the paper.

3.A Derivation of probability of detection

A derivation of the probability of detecting a single OI after a single time-step is given. In the following derivation, the measurement subscript j is dropped; measurements of other objects have no effect on the jth object since objects are assumed independent.

$$P(\mathcal{O}_j^k = 1 | \boldsymbol{z}^{1:k-1}, \boldsymbol{u}^{1:k-1}) \tag{18}$$

Using the law of total probability we obtain

$$\frac{P(\mathcal{O}_j^k = 1, \boldsymbol{z}^{1:k-1}, \boldsymbol{u}^{1:k-1})}{p(\boldsymbol{z}^{1:k-1}, \boldsymbol{u}^{1:k-1})} \tag{19}$$

Focusing just the numerator, we can un-marginalize the location of the robot and the location of the untracked OI.

$$\int_{\boldsymbol{x}^k \in \mathcal{X}^k} \int_{a_j^k \in \mathcal{A}_j^k} p(\mathcal{O}_j^k = 1, \boldsymbol{z}^{1:k-1}, \boldsymbol{u}^{1:k-1}, x^k, a_j^k) d\boldsymbol{x}^k da_j^k = \int_{\boldsymbol{x}^k \in \mathcal{X}^k} \int_{a_j^k \in \mathcal{A}_j^k} p(\mathcal{O}_j^k = 1 | \boldsymbol{z}^{1:k-1}, \boldsymbol{u}^{1:k-1}, \boldsymbol{x}^k, a_j^k) * p(\boldsymbol{z}^{1:k-1}, \boldsymbol{u}^{1:k-1}, \boldsymbol{x}^k, a_j^k) d\boldsymbol{x}^k da_j^k \tag{20}$$

The first multiplicative likelihood, $p(\mathcal{O}_j^k = 1 | \boldsymbol{z}^{1:k-1}, \boldsymbol{u}^{1:k-1}, \boldsymbol{x}^k, a_j^k)$, term is independent of previous object observations and previous control inputs. Thus Eq. (20) is equivalent to:

$$\int_{\boldsymbol{x}^k \in \mathcal{X}^k} \int_{a_j^k \in \mathcal{A}_j^k} p(\mathcal{O}_j = 1 | \boldsymbol{x}^k, a_j^k) * p(\boldsymbol{z}^{1:k-1}, \boldsymbol{u}^{1:k-1}, \boldsymbol{x}^k, a_j^k) d\boldsymbol{x}^k da_j^k = \int_{\boldsymbol{x}^k \in \mathcal{X}^k} \int_{a_j^k \in \mathcal{A}_j^k} p(\mathcal{O}_j = 1 | \boldsymbol{x}^k, a_j^k) p(\boldsymbol{x}^k | \boldsymbol{u}^{1:k-1}) * p(\boldsymbol{z}^{1:k-1}, \boldsymbol{u}^{1:k-1}, a_j^k) d\boldsymbol{x}^k da_j^k \tag{21}$$

Notice that the second multiplicative likelihood term, $p(\boldsymbol{x}^k | \boldsymbol{u}^{1:k-1})$, is the predictive robot distribution. Continuing, (21) is equivalent to:

$$\int_{\boldsymbol{x}^k \in \mathcal{X}^k} \int_{a_j^k \in \mathcal{A}_j^k} p(\mathcal{O}_j = 1|\boldsymbol{x}^k, a_j^k) p(\boldsymbol{x}^k|\boldsymbol{u}^{1:k-1}) * p(a_j^k|\boldsymbol{z}^{1:k-1}, \boldsymbol{u}^{1:k-1}) p(\boldsymbol{z}^{1:k-1}, \boldsymbol{u}^{1:k-1}) d\boldsymbol{x}^k da_j^k \tag{22}$$

Noticing that the final multiplicative term is independent of the integration, and that it cancels with the denominator in Eq. (19), Eq. (3) gives the result ■.

3.B Proof of Prop. 4.1 and constraint properties

Proof: Given Eqs. (15), 16 an expression in terms of $\bar{x}_i^k$ for the following set is desired:

$$\{\bar{x}_i^k | P(\mathcal{O}_{i,j}^K = 1 | z_j^{1:k-1}, \mathrm{u}_i^{1:k-1}) \geq 1 - \alpha\} \tag{23}$$

Using Eqs. (3),(15),(16) the LHS of the condition in (23) is a function of $\bar{x}^k$, and is proportional to a GM distribution. Since the tracked object is represented by a single Gaussian, the expression reduces to:

$$\left\{ \bar{x}_i^k \Big| \sum_{l=1}^{n_s} \zeta_l \cdot \mathcal{N}_{\bar{x}_i^k}\left(c_l + \bar{a}_j^k, \Sigma_{O_l} + \Sigma_i^k + \Sigma_j^k\right) \geq 1 - \alpha \right\}$$

By taking a log, and using Jensen's inequality the following condition produces an inner bound for (23):

$$\sum_{l=1}^{n_s} \ln\left(\frac{\zeta_l}{|2\pi\left(\Sigma_{O_l} + \Sigma_{x_i^k} + \Sigma_{a_j^k}\right)|^{1/2}}\right) - \frac{1}{2} * (\bar{x}_i^k - c_l - \bar{a}_j^k)' \left(\Sigma_{O_l} + \Sigma_i^k + \Sigma_j^k\right)^{-1} (\bar{x}_i^k - c_l - \bar{a}_j^k) \geq \ln(1-\alpha) \tag{24}$$

Let $\tilde{\Sigma} = \Sigma_{O_1} + \Sigma_i^k + \Sigma_j^k$, if $n_s = 1$, then:

$$(\bar{x}_i^k - c_1 - \bar{a}_j^k)'\tilde{\Sigma}^{-1}(\bar{x}_i^k - c_1 - \bar{a}_j^k) \leq -2\ln\left(\frac{(1-\alpha)|2\pi\tilde{\Sigma}|^{1/2}}{\zeta_1}\right) \tag{25}$$

Thus $\mathcal{S}$ is defined by Eq. (25), a convex quadratic in $\bar{x}_i^k - \bar{a}_j^k$ ■.

Notice that if the right hand side of (25) is negative, the tracking constraint cannot be satisfied and the problem is infeasible.

Since c_l have so far been arbitrary in the more general case of (24), there is no formula for the set which satisfies the inequality in terms of $\bar{x}^k$. Sufficient properties for the general GM sensor modal are now shown.

Corollary 3.B.1 *Suppose the sensor is modeled as a quasi-concave GM, and that tracked and agent distributions are Gaussian. If there exists some $\bar{x}^*$ which maximizes Eq.* (10) *and is feasible for $\alpha = 0$, then the set defined by Eq.* (10) *has a subset $\mathcal{S}$ which is convex and non-empty for $\alpha \in [0, \alpha_{\max}]$, $\alpha_{\max} < 1$.*

Proof: Corollary (3.B.1) is a direct consequence of the properties of quasi-convex functions ■.

The constraint defined in (24) is an under-bound on the true constraint set and is much simpler to compute. Notice that the function the the LHS in of Eq. (24) has similar properties to the softmax function, where the largest argument tends to dominate. Roughly speaking, a sufficiently clustered set of mixand means c_l should ensure that (24) has a a solution and can be used instead of Eq. (10).

3.C The assignment problem

In this formulation, in order to guarantee tracking performance the algorithm first assigns at least one robot to each tracked OI. To do this, a variant of the linear assignment problem can

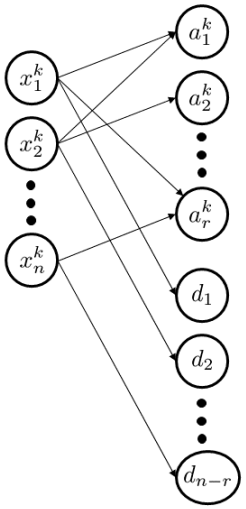


Figure 3.4: The Assignment Problem. Some connections are omitted for clarity.

be solved. First define a mathematical graph (Fig. 3.4) with edges between robot nodes and OI nodes. In addition provide $n - r$ fictitious nodes for each of the $n - r$ robots which will remain un-assigned; here r is the cardinality of $\mathcal{T}$.

The graph only connects robots to objects which can be reached within T. The weight of these corresponding edges is simply the euclidean distance between them. In addition, each robot is connected to the $n - r$ dummy nodes with cost zero.

The given setup is really a simplification of the true underlying assignment problem. As it has been stated above, this problem allows for a fast solution using the Hungarian algorithm or its variants. It does not take into account information gain differences from assignment. In addition, it does not capture behaviors such as "double teaming" an OI to ensure observation. Finally, it does not take into account either robot or OI uncertainties. This may be significant when considering that an OI with low uncertainty may be un-tracked some time while uncertain OIs must be tracked immediately. At the same time, the closest robot to a well known OI may gain more information by switching OIs.

CHAPTER 4

SECURE MINIMUM TIME PLANNING UNDER ENVIRONMENTAL UNCERTAINTY

4.1 Introduction

Automation has permeated many physically consequential tasks as far ranging as self driving cars [2] and nuclear centrifuge control [54]. This ubiquity has increased the need for security in Cyber Physical Systems (CPS). Knowledge of the environment is paramount for autonomous CPS to succeed in many tasks. This knowledge is typically formulated via sensors, perception, and external databases. A CPS must be able to trust its environmental perceptions, or else account for the possibility that the knowledge of its environment is incorrect.

To exemplify the possibility of adversarial attack on vehicles, security flaws have been shown in modern cars. Checkoway *et al.* showed that the control and auxiliary systems of a car can be compromised by a Man In the Middle (MIM) attack where a hacker gains access to a communication network providing the vehicle with outside information [55]. In addition, dangers posed by security flaws are exemplified by several high profile instances such as the STUXNET worm and the reported downing of a U.S. RQ-170 drone in Iran [56]. The STUXNET worm was successful in causing significant damage to the nuclear enrichment program of Iran, while the RQ-170 event exemplifies the kinds of dangers which must be addressed if large scale automation of vehicles is to become a reality.

To understand the some of the security challenges in CPS, consider a self driving car. Traditionally, knowledge of the environment, such as a map, is provided as a polygonal or occupancy-grid representation [19]. Many modern self driving frameworks provide this information via a networked server [54], which can be vulnerable to cyber attack. If a server is vulnerable, the autonomous car is unsure if provided information is fully correct, incomplete, or maliciously designed. An abstraction of a dangerous planning scenario is exemplified by the dotted red path in Fig. 4.1, where a point robot is operating in a bounded, static, obstacle environment. The robot plans the fastest path around a known red obstacle, but will not be able to react in time to the unknown obstacle shown in black.

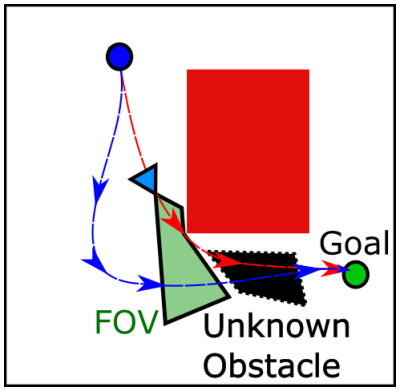


Figure 4.1: The robot (blue triangle) seeks to achieve the goal position (green circle). The sensor FOV is in light green, but is obstructed by the known obstacle (red) . An unsafe path (red) skirts around the obstacle. A safe (blue) path gives wide berth to the known obstacle and has time to avoid the unknown obstacle (black).

The goal of this work is to create an optimal planner which is robust to the lack of environmental knowledge (map modifications), and is motivated by the security risks in CPS exemplified above. The approach utilizes trajectory generation and direct collocation methods to produce dynamically aware trajectories which guarantee obstacle avoidance while producing minimum time paths. Unlike previous work, which uses pre-defined motion primitives and can be restrictive in cluttered environments, the formulation presented here utilizes geometric concepts to formulate analytic constraints which enable the trajectories to be generated in real time using sequential quadratic programming.

4.2 Previous Work

Previous work on control and planning security for CPS typically fall into two primary categories: detecting Man in the Middle (MIM) attacks, and operating robotic systems under environmental uncertainty.

Most works focusing on attack detection assume that data provided to the controller (measurements) are compromised. Mo, *et al.* consider detection of playback attacks on a Linear Time

Invariant (LTI) system, and propose a sub-optimal noisy control to increase attack detection rates [57]. Pasqualetti *et al.* provide detailed analysis of *detectability* and *identifiability* of noiseless LTI systems [58]. Fawzi *et al.* provide theoretical bounds on how large the support of an output attack vector can be while still ensuring that the attack can be detected [59]. An NP-hard detector is also provided in the form an l_0 minimization, and a tractable l_1 approximation. Pajic *et al.* utilize a similar method for noisy dynamical systems [60]. Shoukry *et al.* focus on generating a *sound* method using *Satisfiability Modulo Theories* to ensure that a correct estimate of the attack vector is returned [61].

The work presented here relates the recent and on going work on the effects of cyber attacks on sensing, presented above, to the problem of global planning under an unknown map, which is somewhat traditional. Unlike the work presented here, this traditional problem usually makes a kinematic assumption where the robotic platform is able to stop instantaneously. Several recent surveys detail much of the literature in this area [62, 63]. To the authors' knowledge, Shkel and Lumelsky first considered the effects of dynamics and inertia on the planning problem [3]. They dubbed this the "Jogger's Problem" since it was analogous to a jogger making their way through unknown terrain, while planning their motion within the instantaneous Field Of View (FOV). This classical approach uses a circular FOV and known 'stopping' paths. Lopez and How extend the Jogger's Problem for FOVs appropriate to quad-copters [64], generalizing stopping paths to 3D, and solve the planning problem in real time through the use of motion primitives and sampling. In all current work on the Jogger's Problem, paths are planned using the instantaneous FOV. Both the circular FOV assumptions as well the utilization of an instantaneous FOV for path planning are highly restrictive. More specifically, utilizing an instantaneous FOV is a critical limitation for sensors which are not circular. This includes the majority of modern sensors such as cameras and laser range finders which have limited angular extent.

Several recent studies consider robotic path planning with environmental uncertainty. Missiuro and Roy adapted a Probabilistic Random Map (PRM) to account for uncertainty in a SLAM

map [65] and use linear interpolation of uncertainty ellipsoids along obstacle edges to give a probabilistic guarantee on collision avoidance. Their work assumes that the number shape and nominal size of obstacles is already known and that a robot could recover from a crashed state. Similarly, Vitus and Tomlin study uncertainty in obstacle vertices [66], and provide chance constraints on obstacle avoidance. A hybrid analytic-sampling method is proposed which reduces complexity compared to previous techniques. In both of these works, the environment's shape is fully known, but certain details are uncertain and they do not account for robotic dynamics.

In summary, current literature addresses the detection and estimation of cyber attacks on linear systems, or considers operation with noisy sensors or adversarial pursuers. Conversely, the effect of a mapping attack has not been considered. Mapping attacks are fundamentally different from sensing attacks since sensor models are usually well known and accurate. In addition, the traditional approaches to the Jogger's problem provide very conservative motions by either using circular or instantaneous FOV. The primary contribution of this work is an algorithm which provides guaranteed collision-free planning for a robotic system with limited or compromised knowledge of its environment. This algorithm relaxes the visibility constraints traditionally used in the Joggers problem and reactive planning. The continuous time problem is formulated and solved using direct optimization methods through a Nonlinear Program (NLP). The framework allows generalization of the approach to a variety of robotic system models, such as quad copters and other ground vehicles.

4.3 Problem Description and Definitions

The problem of dynamic planning under environmental uncertainty is first described qualitatively. Formal definitions are then given, and a mathematical formulation is provided. The problem formulation considers a 3D Euclidean configuration space of a point robot; the results are demonstrated for the 2D case in Section 4.7.

Qualitatively, the problem statement is 'Plan a path from a start point s to an end point e in minimum time, and guarantee that the robot avoids collisions due to potentially incorrect environmental knowledge.' Throughout this work, the robot is assumed to desire an optimal trajectory, but is equipped with a control strategy which allows it to avoid detected obstacles that were not known *apriori*. The intuition behind this two tiered method is that optimal trajectory planning is computationally expensive relative to state based feedback control, and re-planning in the face of new obstacles may not be fast enough to avoid such obstacles. In addition, state based feedback control can make guarantees on steady state behavior that trajectory generation usually cannot. Several standard assumptions are required to make this problem well posed. First, a dynamically feasible path is assumed to exist between s and e, both in the true environment, $\mathcal{C}_{\text{free}}$, as well as in the robot's current knowledge of the environment, $\hat{\mathcal{C}}_{\text{free}}$. Second, the robot state at time t can be represented by a point $x(t) \in \mathbb{R}^n$, with a sensor field of view $S_s(x) \subset \mathbb{R}^3$, which varies only with robot state and intersections with obstacles in $\mathcal{C}_{\text{obst}}$. Finally, sensor data and control inputs are assumed un-compromised.

In the following formulation, $J(\cdot)$ is the cost function, which in Lagrange form is the integral of the Lagrangian $\mathcal{L}(x(t), u(t), t)$ up to the final time t_f. The configuration space, $\mathcal{C} = \mathcal{C}_{\text{obst}} \cup \mathcal{C}_{\text{free}}$, is a union of the 'free' space and 'obstacle' or occupied space, and $u(t) \in U \subset \mathbb{R}^m$ are controls. A Lagrange formulation for the optimal control problem, with perfect state information and incomplete environmental knowledge, is described by:

$$\begin{aligned} \underset{u(t)\in U, t_f}{\text{minimize}} \quad & J(x(t), u(t), t_f) \\ s.t. \quad & \dot{x}(t) = f(x(t), u(t)) \\ & g(x(t), u(t)) \leq 0 \\ & x(t) \in \hat{\mathcal{C}}_{\text{free}} \quad \forall t \in (0, t_f) \\ & x(0) = s, \quad x(t_f) = e \end{aligned} \tag{1}$$

This differs from the classical optimal control problem because the robot does not know $\mathcal{C}_{\text{free}}$, but instead knows some unverified estimate, $\hat{\mathcal{C}}_{\text{free}}$, and must plan over $\hat{\mathcal{C}}_{\text{free}}$. There are many approaches to planning over an unknown space which include probabilistic approaches seeking to optimize some mean performance (expectation) as well as robust approaches which seek to find a solution for the worst-case. This work can be viewed as a form of 'robust' path planning which deterministically plans over $\hat{\mathcal{C}}_{\text{free}}$. Here the word robust is not used in the control theoretic sense, which defines robustness with respect to a disturbance, but rather with respect to unknown changes in the map. Since $\mathcal{C}_{\text{free}}$ is unknown, it is prudent to consider the interaction of the FOV $S_s(x)$ and $\mathcal{C}_{\text{obst}}$.

Definition 4.3.1 An obstacle $O \subset \mathbb{R}^3$ is *visible* at time $t \iff O \cap S_s(x(t)) \neq \emptyset$

This definition of visibility does not imply that the shape of O is known after one such observation. Rather, it simply states that a part of the obstacle is withing the FOV of the robot at t and is not obstructed by other obstacles in the environment.

If a robot encounters an unknown obstacle (i.e. it becomes partially visible), the robot may need to react to avoid the obstruction. In this work, it is assumed that the robot uses the abstract planning approach described by Algorithm (4). The planner solving line 1 is usually dubbed the path planner, while the evasive controller using $\pi(x(t), \hat{\mathcal{C}}_{\text{free}})$ is usually called a 'low level' controller. The combination of these two control methods forms a complete strategy, where the robot utilizes the optimal trajectory unless avoidance behavior is necessary. The formulation presented here does not take into account errors in positional estimation or sensor noise, although the methods presented here can be extended to include bounded disturbances by using reachability analysis like those described in [67].

Clearly the ability of the robot to intelligently evade an unknown obstruction, line 7 in Algorithm (4), must be considered. The intuition behind the sequel comes from the wealth of

research which has analyzed reactivity to unknown or moving obstacles in the configuration space [63, 68]. In the formulation presented here, a control law which reacts to an obstacle is called a 'reactive controller'. Conceptually, when a robot encounters an unknown static obstacle in the environment, new information is gained about the configuration space which may make the previously planned path infeasible or sub-optimal. Consider a locally optimal solution $x^*(t)$ to (1). Recall that the robot plans over $\hat{\mathcal{C}}_{\text{free}}$ instead of the true $\mathcal{C}_{\text{free}}$. If $x^*(t) \subset \mathcal{C}_{\text{free}}$, this means that a path is safe, but perhaps sub-optimal. In other words, even if $\hat{\mathcal{C}}_{\text{free}}$ is incorrect and some unknown obstacle $O \nsubseteq \hat{\mathcal{C}}_{\text{obst}}$ is visible at t, the robotic system need not react to this new information because collision is not imminent. Conversely, if $x^*(t)$ passes through such an obstacle, the robot must react. A key challenge is that a practical robotic system has dynamics and cannot stop instantaneously. Thus, if a planned path collides with an unknown obstacle, the robot must observe the obstacle far enough in advance to avoid collision, where 'far enough' depends on the dynamics and inertia of the particular platform.

Note that this work does not prescribe a particular method for reacting to unknown obstacles in the environment. Instead, a reactive controller is assumed to exist with a control law, $\pi(x(t), \hat{\mathcal{C}}_{\text{free}}) \rightarrow U$, which is only a function of $x(t)$ and the current estimate of the environment. Let, $\mathcal{M}$ be the set of 3D non-empty polyhedra (obstacles). The true map, $\mathcal{C}_{\text{obst}}$, is assumed to be well approximated by a union of a finite number of such polyhedra. The reactive control law $\pi(\mathbf{.})$ is assumed to ensure $x(t)$ remains bounded near the position where the robot first encounters an unknown obstacle, i.e. the system is stable in the sense of Lyapunov. Note that a control law which ignores knowledge of the current map estimate $\hat{\mathcal{C}}_{\text{free}}$, and thus ignores the possibility of collision, is a special case: i.e. $\pi(x(t))$. Most robotic systems have Lyapunov stable control laws which do not depend on $\hat{\mathcal{C}}_{\text{free}}$. For example, car-like vehicles are passively stable, rotor craft and underwater vehicles can hover (neutral buoyancy), and fixed wing vehicles can exhibit loitering behavior [69].

Input: $e, s, x(0), \hat{\mathcal{C}}_{\text{free}}$
1 $[x^*(t), u^*(t)] \leftarrow$ Solve Eq. (1);
2 **while** $\mathcal{O}_u \cup S_s(t) = \emptyset$ **do**
3 $\quad u(t) = u^*(t)$
4 **end**
5 **if** $x^*(t)! = safe$ **then**
6 $\quad$ **while** *Evasive maneuver* **do**
7 $\quad\quad u(t) = \pi(x(t), \hat{\mathcal{C}}_{\text{free}})$
8 $\quad$ **end**
9 **end**
10 **go to** 1

Algorithm 4: Robotic Planning Algorithm

Since $\pi(x(t), \hat{\mathcal{C}}_{\text{free}})$ is stable, it helps define a *reactive set*. While the framework presented here holds for more general policies, for the remainder of this exposition, the control law π is assumed to depend only on the state of the robot. Let the reactive path produced by π be $x_r(t, \pi(x(t)))$ with initial condition $x(\tau)$.

Definition 4.3.2 The reactive set $S_r(x(\tau))$ defined by control law π and the initial condition $x(\tau)$ is:

$$S_r(x(\tau)) := x_r(t, \pi(x(t))), \quad t \in [\tau, \infty), S_r(x(\tau)) \subset \mathbb{R}^3$$

In the above definition, and this paper, the reactive set and FOV are subsets of the configuration space rather than the full state space. Since the robot is assumed to be a point, collisions are avoided *iff* the 3D position of the robot is outside of obstacles. Similarly, the FOV is contained in $\mathbb{R}^3$ since it deals exclusively with sensing the physical environment. This assumption can be relaxed by increasing the size of obstacles in $\hat{\mathcal{C}}_{\text{free}}$ by a circular over-approximation of the robot's shape and reducing the problem to the one presented here.

The reactive set can be considered as a generalization of stopping distance. To give intuition to this definition, consider a one dimensional problem where a robot has dynamics $\dot{x}_1(t) = x_2(t)$, with bounded acceleration control $\dot{x}_2(t) = u(t)$, $u \in [-1, 1]$; the robot's state is position and velocity. In this case, an obstacle is a value on the real line and a reactive controller could simply

try to stop the vehicle as quickly as possible:

$$\pi(x(0)) = \begin{cases} 1 & x_2(t) < 0 \\ -1 & x_2(t) > 0 \\ 0 & o.w. \end{cases}$$

Since the physical position of the robot is one dimensional, the reactive set is then the interval determined by the initial position and stopping distance of the robot: $S_r(x(\tau)) = [x_1(\tau), x_1(t_r)]$, where $t_r \geq \tau$ is the first time the robot is at rest.

The reactive set, as defined here, is a closed-loop invariant set of the robotic system restricted to the physical space. This concept is related to reachable sets which consider noisy inputs. Although disturbances are not explicitly considered in this work, calculating reachable sets which include disturbances can allow the methods proposed here to be robust, in a control theoretic sense, against bounded disturbances [70]. Most reachability analysis has focused on linear systems, and good results have been obtained for reasonably sized linear systems [67]. Recent work has provided methods for accurate over-approximations of reachable sets of non-linear systems [71].

4.4 Planning under adversarial maps

Several arguments are necessary to motivate a solution to the problem of planning under adversarial environmental attack. The first shows that, in order to guarantee planning under adversarial attack, the reactive set $S_r(x(t))$ must be fully observed at some previous time. Second, given that $S_r(x(t))$ is guaranteed to be observed along a trajectory, one needs only ensure that $S_r(x(t)) \cap \hat{\mathcal{C}}_{\text{obst}} = \emptyset$ along a path $x(t)$ to ensure that the robot can avoid any unknown obstacle.

Given that the robot is planning over $\hat{\mathcal{C}}_{\text{free}}$, it is necessary to consider the effects of the mis-

match between $\hat{\mathcal{C}}_{\text{free}}$ and the true $\mathcal{C}_{\text{free}}$ on safety. The portion of $\mathcal{C}_{\text{free}}$ which is incorrectly assumed to be obstructed is inconsequential in terms of safety, i.e. $\hat{\mathcal{C}}_{\text{obst}} \cap \mathcal{C}_{\text{free}}$. The 'dangerous' mismatch is described by the set of unknown obstacles $\mathcal{O}_u \coloneqq \mathcal{C}_{\text{obst}} \cap \hat{\mathcal{C}}_{\text{free}}$. When the robot senses a component of $\mathcal{O}_u$ which intersects with its planned path and switches to a reactive controller, the robot must have a guarantee that the evasive maneuver will not cause an obstacle collision. By definition, $S_r(x(0))$ contains such evasive maneuvers. Since $\mathcal{O}_u$ is unknown, the only way to guarantee safety is to ensure that the robot has previously observed its planned reactive set. In other words, the reactive set at any time τ must be contained within he observed area up to that point:

$$\bigcup_{t\in[0,\tau]} S_s(x(t)) \supseteq S_r(x(\tau)) \tag{2}$$

The final argument requires one more definition.

Definition 4.4.1 A planned path $x(t)$ defined on $t \in [0, t_f]$ is *safe* if it satisfies the constraint (2) and $S_r(x(t)) \cap \hat{\mathcal{C}}_{\text{obst}} = \emptyset$

Suppose that the visibility constraint (2) is satisfied for a particular trajectory $x(t)$, and assume that the robot continuously updates its estimate of the environment $\hat{\mathcal{C}}_{\text{free}}$ and $\hat{\mathcal{C}}_{\text{obst}}$. In practice, this implies maintaining an occupancy grid or 3D occupancy representation such as OctoMap [19].

Theorem 4.4.2 *Let $x^*(t)$ be a safe, planned, dynamically feasible trajectory defined on $t \in [0, t_f]$. Suppose the robot utilizes a reactive controller $\pi(x(t))$ and Alg.4. If the first unknown obstacle is seen at τ, then any realized robotic path $x(t) = \{x^*(t) | t \in [0, \tau), x_r(t, \pi(t)) \quad o.w.\}$ will be collision free.*

Proof: The proof proceeds by construction. Consider the first instance in time $\tau \in [0, t_f]$

when an unknown obstacle $O \subset \mathcal{O}_u$ is visible. A reactive controller is only be utilized if O causes $x^*(t)$ to no longer be safe. At τ, the obstacle space estimate is updated to be

$$\hat{\mathcal{C}}_{\text{obst}} \leftarrow \big(\hat{\mathcal{C}}_{\text{obst}} \cup O\big) / \big(\bigcup_{t\in[0,\tau]} S_s(x(t)) \bigcap \mathcal{C}_{\text{free}} \big).$$

and the free space estimate is updated to be

$$\hat{\mathcal{C}}_{\text{free}} \leftarrow \mathcal{C} / \hat{\mathcal{C}}_{\text{obst}}$$

By definition, $x^*(\tau) \in S_r(x^*(\tau))$, therefore, if $x^*(t), t \in [\tau, t_f]$ remains safe under the updated $\hat{\mathcal{C}}_{\text{free}}$, the planned trajectory can continue to be executed without re-planning and without collision. Conversely, if $x^*(t)$ is no longer safe, the reactive control policy $\pi(x(t))$ is utilized. By definition, the reactive controller with initial condition $x^*(\tau)$ satisfies $x_r(t, \pi(x(t))) \subset S_r(x^*(\tau))$. Because constraint (2) is satisfied, and τ is the first time a component of $\mathcal{O}_u$ is visible, it must be that $S_r(x^*(\tau)) \subset \mathcal{C}_{\text{free}}$■.

Theorem (4.4.2) says that, to guarantee safety, one need only ensure the reactive sets, $S_r(x(t))$, have been observed in advance and that they do not intersect with any known obstacles: $O \subset \hat{\mathcal{C}}_{\text{obst}}$. Theorem (4.4.2) motivates the final formulation of the optimal planning problem under map uncertainty:

$$\begin{aligned}
\underset{u(t)\in U, t_f}{\text{minimize}} \quad & J(x(t), u(t), t_f) \\
s.t. \quad & \dot{x}(t) = f(x(t), u(t)) \\
& S_r(x(t)) \subset \bigcup_{\tau\in[0,t]} S_s(x(\tau)) \\
& S_r(x(t)) \subset \hat{\mathcal{C}}_{\text{free}} \quad \forall t \in (0, t_f) \\
& x(0) = s, \quad x(t_f) = e
\end{aligned} \tag{3}$$

Note that the only difference between the formulation above and that in (1) is replacement of the

path constraints $g()$ as set constraints which guarantee obstacle avoidance though the visibility of the reactive set and its inclusion in $\hat{\mathcal{C}}_{\text{free}}$. The problem must be refined further by reducing these set constraints to be of the form of $g()$ which necessitates some simplifications.

4.5 Approximating the Set Constraints

The set inclusion constraints in (3) are intuitive, but difficult to compute in practice. Conversely, if the set constraints can be written as path inequality constraints of the form in (1), trajectory optimization using direct collocation methods can be used to solve the optimal control problem.

4.5.1 The Visibility Constraint

Definition (4.4.1) requires that the entire reactive set $S_r(x(t))$ is observed by the time the robot is at $x(t)$. A tighter constraint is:

$$S_r(x(t)) \subset S_s(x(\tau)), \quad 0 < \tau < t \tag{4}$$

In other words, the reactive set at time t must be seen, in its entirety, at some particular previous time instance. Therefore, a solution to the following problem would also be a solution to problem (3):

$$
\begin{aligned}
\underset{u(t)\in U, t_f}{\text{minimize}} \quad & J(x(t), u(t), t_f) \\
s.t. \quad & \dot{x}(t) = f(x(t), u(t)) \\
& \exists \tau < t \quad s.t. \quad S_r(x(t)) \subset S_s(x(\tau)) \\
& S_r(x(t)) \subset \hat{\mathcal{C}}_{\text{free}} \quad \forall t \in (0, t_f) \\
& x(0) = s, \quad x(t_f) = e
\end{aligned}
\tag{5}
$$

Note that this constraint generalizes of the instantaneous FOV planning utilized for the classical Jogger's Problem and such algorithms as [64], since it does not require a circular or spherical safe stopping set, and is a contribution of this work.

4.5.2 The Reactive Set

To achieve a practical solution, an upper bound is proposed for the reactive set. This work uses an ellipsoidal upper bound to approximate $S_r(\centerdot)$. There are several properties which make an ellipsoidal approximation appealing. First, bounding simulated or observed trajectory data with ellipsoids is relatively straightforward using least squares optimization. Second, ellipsoids are easily manipulated to provide constraint equations of the form in problem (1). Finally, if a direct collocation method is utilized to perform the discretization and optimization of the optimal control problem, the constraint $S_r(x(t)) \subset \hat{\mathcal{C}}_{\text{free}}$ can be formed into a state-varying distance condition. Details on how to practically perform this approximation are given in Section 4.7 and [72].

Let $Q(\centerdot)$ be a real, positive-definite, diagonal, and continuously varying matrix. Let $R(x)$, $c(x)$, and $a(x)$ be the rotation matrix, position of the robot, and ellipse center respectively. Note that $c(x)$ is simply a subset of the elements of x corresponding to the 2 or 3D robot position. For the remainder of this work, $S_r(\centerdot)$ is assumed to take the form:

$$S_r(x) = \{y = R(x)(\sqrt{Q(x)}z + a(x)) + c(x) | z^T z \leq 1\} \tag{6}$$

Clearly Q and a depend on x since the stopping distance of an inertial object depends on its initial momentum.

4.6 Discretizing the Problem

Direct trajectory optimization methods have seen wide acceptance in recent robotic literature and practice [49], and can find locally optimal solutions even in the face of non-linear and non-convex constraints. Their appeal comes from recent improvements in optimization tools and their wide availability [52]. In addition, direct methods require less expert knowledge in optimal control when compared to methods based on the calculus of variations, dynamic programming, or fast marching [49].

The above analysis motivates the use of a direct optimization method for the problem presented in this work since the constraints proposed are complex (nonlinear, time-varying) and vary with the model of the FOV and $S_r(\mathbf{.})$. This section presents a Nonlinear Program (NLP) discretization of problem (5). The connection between this NLP and (5) is not direct; additional details are provided in the appendix. Finally, a model of the sensor set $S_s(x)$ is presented and the set inclusion constraints in (5) are converted into inequality constraints.

4.6.1 Nonlinear program formulation

The formulation in (5) differs from standard trajectory optimization problems only in its set constraint requiring visibility and non-collision of $S_r(x(t))$. Reference [49] provides details on

standard discretizations of the objective function, dynamic, and obstacle constraints. This section presents details on the set inclusion constraint.

Let x_i be the ith discretized state, where $x_i := x(t_i)$. Following the notation in [49], the ith dynamics defect is denoted ζ_i. In addition $L(\mathbf{.})$ is the discretized, numerically integrated form of $\mathcal{L}(\mathbf{.})$. The visibility constraint can be discretized by requiring that $S_r(x_i)$ is observed by a previous discretization point with a fixed time index difference:

$$\begin{aligned} \underset{u_i \in U, t_f}{\text{minimize}} \quad & L(x_1, .., x_K, u_1, .., u_K, t_f) \\ s.t. \quad & \zeta_i = 0, d(S_r(x_i), \hat{\mathcal{C}}_{\text{obst}}) > 0 \quad \forall i \\ & S_r(x_i) \subset S_s(x_j) \\ & x_1 = s, \quad x_K = e \\ & j = i - \Delta i, i \in \{\Delta i, .., K\} \end{aligned} \tag{7}$$

where the distance $d(\mathbf{.}, \mathbf{.})$ is Euclidean when referring to points and Hausdorff between sets in $\mathbb{R}^n$, where n corresponds to the appropriate dimension.

Since Δi is fixed and the union of sets has been ignored, $S_r(x_i) \subset S_s(x_j)$ is a tight discrete approximation of the set inclusion constraint in Eq. 5.

4.6.2 Sensor model and visibility constraint

The visibility constraint in (7) must be converted into a set of inequalities. First, consider the bijective transformation $\mathcal{F} : \mathbb{R}^3 \to \mathbb{R}^3$ defined by the property: $\mathcal{F}(S_r(x_i)) = B_0(1)$. In other words, it takes the ellipsoidal reactive set to the unit ball. If the sensor FOV is modeled as a polyhedra, represented analytically as set of inequalities, then applying $\mathcal{F}$ to the FOV will result in another set of inequalities. In addition, since the obstacles are assumed to take the form of

a finite set of polyhedra, $\mathcal{F}$ can also be utilized to perform fast collision checking and distance calculations.

In this work, the sensor FOV is represented by a subset of $\mathbb{R}^2$ or $\mathbb{R}^3$, depending on the representation of the environment. In 2D, many sensor models such as LIDAR can be modeled as a 'slice' of a disc with the radius being the effective sensor range r. A more conservative approximation is an isosceles triangle with equal sides of length r. A similar assumption is made for cameras in $\mathbb{R}^3$ using the pinhole-camera assumption and assuming a rectangular digital image sensor. Consider the case when no obstacles are in the FOV of the robot, then the set $S_s(x)$ is a polyhedron in the appropriate dimension, and can be expressed through the linear inequalities.

$$S_s(x) = \{y | A(x)y \leq b(x)\} \quad y \in \mathbb{R}^3 \tag{8}$$

Because there are no obstacles, the shape of FOV is not time dependent, and $S_s(x)$ is a rotation and translation of $S_s(0)$. In order to concisely express the reactive set inclusion constraint, it is desirable to transform the sensor set $S_s(x_j)$ via the bijective map $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}^n$ which is defined by property: $\mathcal{F}(S_r(x_i)) = B_0(1)$. In other words the ellipsoidal reactive set at time t_i is mapped to the unit ball. Let $R_i := R(x(t_i))$ be the rotation matrix (in 2D or 3D) of the robot at time t_i. Given the representation of $S_r()$ and $S_s()$, the reactive set constraint in (7) can be written concisely as:

$$\begin{gathered} \forall i \in \{\Delta i, .., K\}, \qquad\qquad j = i - \Delta i \\ A(x_j)(R_i a(x_i) + c(x_i)) \leq b(x_j), \quad d(\tilde{S}_s(x_j, x_i), B_0(1)) \geq 1 \end{gathered} \tag{9}$$

where $\tilde{S}_s(x_j, x_i)$ is $\mathcal{F}(S_s(x_j))$. Equation (9) states that the ellipsoid's center must be within the FOV, and that the translated and skewed FOV's sides must be outside the unit ball. Note that, in the case of polygonal obstacles, the obstacle distance constraint in (7) is computed by transforming obstacles via $\mathcal{F}$ and ensuring that the transformed obstacles' sides are outside the unit

sphere.

One final condition must be considered to fully ensure visibility. Until now, the exposition has not considered occlusions which may occur from obstacles in the FOV of the robot. Figure 4.1 shows such a scenario. Clearly this occlusion shadow must be considered. For the following result, when referring to a robot state x_j it is implied that this is the 2D or 3D robot position, which is a subset of the elements of the full robot state. Notice that if multiple obstacles intersect the FOV at any one time, the resultant FOV may become very complex. Due to the choice of $S_r()$, this complex shape need not be considered. Lemma (4.6.1) states this concisely. For readability, the proof is left to the appendix.

Lemma 4.6.1 *Assume the constraints in* (9) *are met and that* $S_r(x_j) \subset \mathbb{R}^2$. *Let* $x'_j = \mathcal{F}(x_j)$. *Define the triangle* $\Delta x'_j p_u p_d$ *as shown in Fig. 4.2. Then* $S_r(x_i)$ *is not occluded by any obstacle* $\iff d(\Delta x'_j p_u p_d, \mathcal{F}(\hat{\mathcal{C}}_{\text{obst}})) > 0$. ■

Lemma (4.6.1) says that the geometric cone (red triangle) shown in Fig. 4.2, is the minimum set which must be un-obstructed to ensure full visibility of a reactive set (which is itself free of known obstacles). Note that the proof above considers the 2D case. The 3D case is no different, with the exception that instead of a triangle, a circular (geometric) cone is generated by the tangent planes of the unit sphere and x'_j. For practical purposes, this cone should be approximated by a intersection of a finite number of such tangent planes (i.e that it is approximated by a polyhedron) in order to make the computation of distances efficient.

4.6.3 Limitations on π and S_r

One final requirement must be imposed on $S_r(x)$ in order to allow for a computationally efficient solution. $S_r()$ must vary continuously with its argument. If this is not the case, a discontinuous

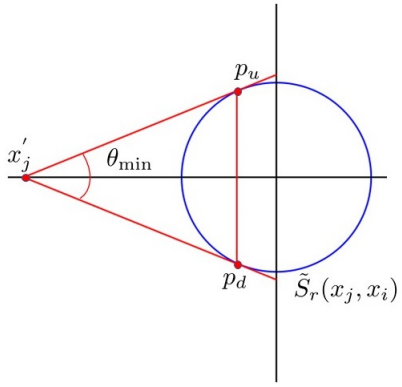


Figure 4.2: A representation of the transformed reactive set. Note $\mathcal{F}$, not explicitly shown here, is unique up to a rotation. Therefore, $x_j^{'}$ can be assumed to lie on the first axis $w.o.l.g$.

constraint function is implied, making the use of modern NLP solvers difficult. To guarantee this behavior, the reactive controller π is required to generate smoothly varying paths with respect to its initial condition. In other words, let $Q(x)$ be the minimal bounding ellipse of the reactive trajectory generated by the policy π with the initial condition x, then:

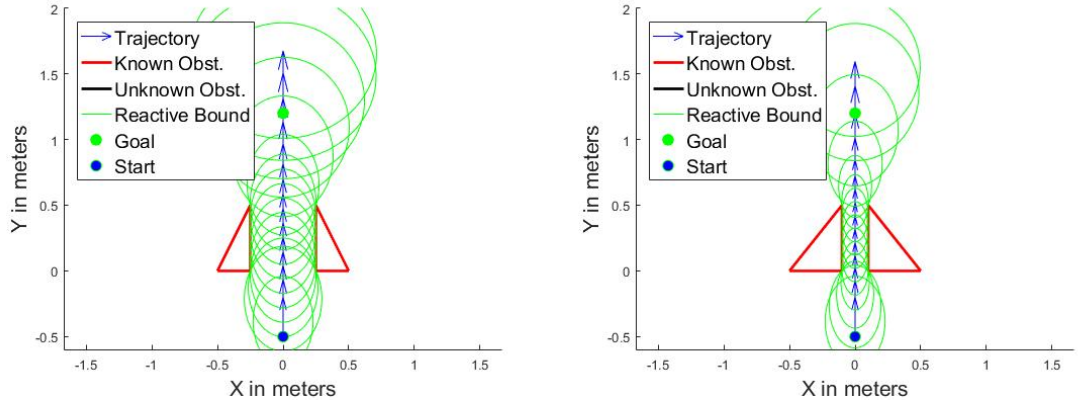
$$\forall \epsilon \quad \exists \delta \quad s.t. \quad \forall \Delta x \in \mathbb{R}^n, ||\Delta x|| < \delta \implies d(Q(x), Q(x + \Delta x)) < \epsilon \tag{10}$$

where the norm of Δx is Euclidean, and the distance d is Hausdorff. This assumption works well in practice, particularly for ground robots, but may be restrictive or conservative in some cases.

4.7 Results & Discussion

4.7.1 Results

Several scenarios are used to exemplify behavior and capability of the secure planning formulation. In the following results the same minimum time objective function is used to compute the trajectories in line 1 of Alg. 4. In addition, the same Model Predictive Controller (MPC) is used

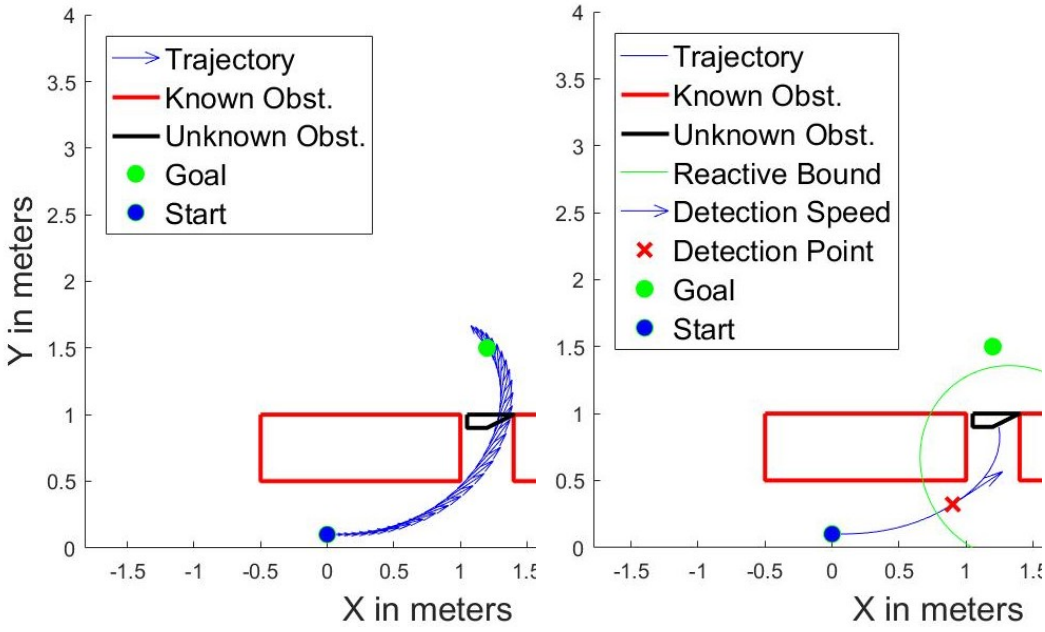


(a) 50 cm wide passage (b) 20cm wide passage

Figure 4.3: Slowing behavior of the secure planner for two passage widths. Arrow size indicates speed and all other parameters are identical.

for the evasive maneuvers in line 7 of Alg. 4. This MPC reactive controller penalizes distance to obstacles and deviation from the position $x(\tau)$. A bounded acceleration differential drive model is utilized to emulate the behavior of ground robots [19]. For details on the exact formulation of both the NLP in (7), the reactive controller, as well as a MATLAB simulator, see [72] and the documentation therein.

In Figure 4.3, the robot is tasked with navigating a narrow passageway with no unknown obstacles of two different widths. The robot starts at position $(0, -0.5)$ oriented along the y axis with an initial velocity of 1m/s. In Figure 4.3(a), the passageway has a width of 50cm. In both cases, robot's initial velocity reduces while in the passageway to ensure that the ellipsoidal reactive set, in green, shrinks to fit within the polygonal constraints. As the robot exits the passageway, it accelerates quickly to reduce the objective function (t_f) in the area with no obstacles. In Figure 4.3(b), the passageway has been narrowed to 20cm. In this case, a much more substantial reduction in speed is noticeable, as denoted by the smaller velocity arrows and their increased density while traversing the passageway. The robot achieves the goal in Figure 4.3(a) in 1.60 seconds while that in Figure 4.3(b) requires 2.12 seconds. As mentioned in section 4.5.2, $S_r(x(t))$ depends primarily on linear and angular velocity (momentum) and actuation limits. This means that, since the robot cannot stop as quickly, it must move more slowly through

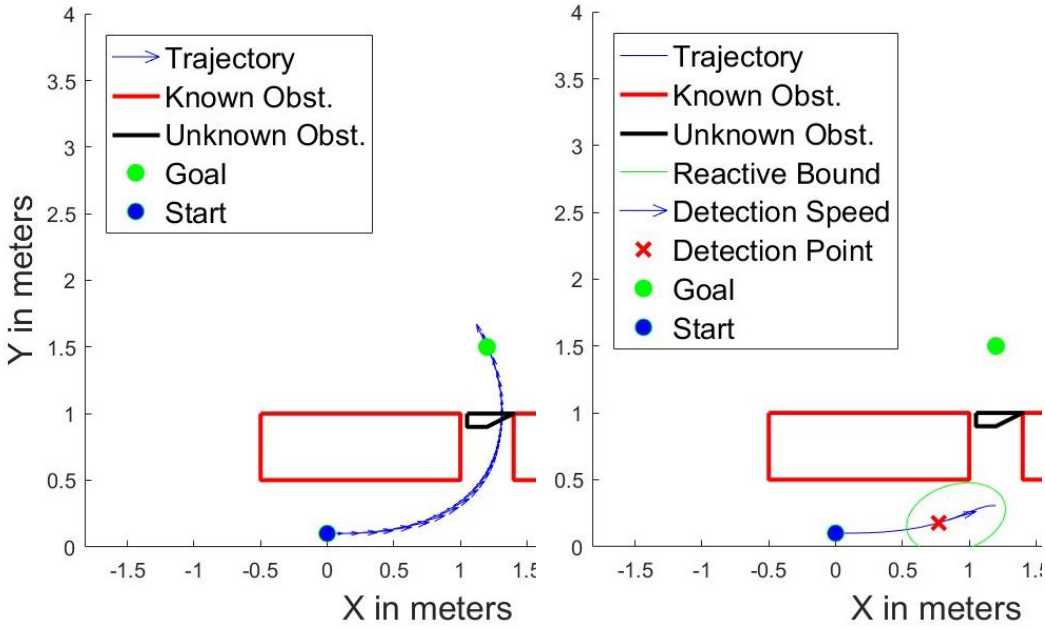


(a) Initial planned trajectory (b) Realized trajectory and collision

Figure 4.4: Behavior of a base-line minimum-time planner, with dynamics constraints. The planned trajectory takes 1.67 seconds, but results in a collision.

narrower passageways in order to sufficiently reduce the size of $S_r(x(t))$.

Figures 4.4 and 4.5 show the guaranteed planner compared to a base-line minimum time planner with obstacle avoidance. In each case, the planners generate a minimum time trajectory; an unknown obstacle is detected as the robot rounds the corner, and the reactive controller is engaged to avoid collision. Figure 4.4(a) shows the planned trajectory of the baseline planner while Figure 4.4(b) shows the realized trajectory. To avoid visual clutter, the sensor FOV is not drawn, but has a range of $r = 2$m and angular range of $\angle 120$. The detection point, red X, denotes the position of the robot when the unknown obstacle is observed: $x(\tau)$. In Fig. 4.4(b), a collision results because the baseline planner did not give the reactive controller enough time to avoid the obstacle. Conversely, Fig. 4.5 shows the respective trajectories for the guaranteed planner. The robot slows down before taking the left-hand turn in order to maintain visibility of its future reactive regions. The robot maintains a much larger distance from the known obstacle (red) and observes the unknown obstacle while initiating its turn maneuver. The reactive controller subsequently activates and successfully brings the robot to a stop.



(a) Initial planned trajectory (b) Realized trajectory and avoidance

Figure 4.5: Behavior of the secure planner. The safe path takes 2.50 seconds.

A C++ implementation was utilized to analyze the real time potential of the method presented here. This experiment was run on a 3.2 GHz Intel Core i7-8700 CPU using the SNOPT Sequential Quadratic Programming (SQP) solver as a back-end [73]. No parallelization was utilized, and all problems were solved on a single core. The scenario in Fig. 4.6 was used to generate five different examples. The generated trajectories are defined by their initial and terminal conditions as follows: A-B, A-C, B-D, C-D, C-E. Additionally the terminal conditions of the initial partial trajectories A-B, and A-C are utilized as initial conditions for trajectories B-D and C-D, C-E respectively. This ensures continuity and dynamic feasibility for each trajectory in its entirety in order to emulate real world behavior. To help analyze the sensitivity of the problem to differing initial conditions, 100 randomly generated initial conditions are evaluated for each partial trajectory using a uniform distribution with support of 0.1 meters in the x-y space and 15 degrees in orientation. Figure 4.7 shows the computational results of this experiment. The histograms in Figure 4.7 use bins of 0.25sec and show distributions run times with a tight variance. This implies that the problem solution is relatively insensitive to moderate variation in its

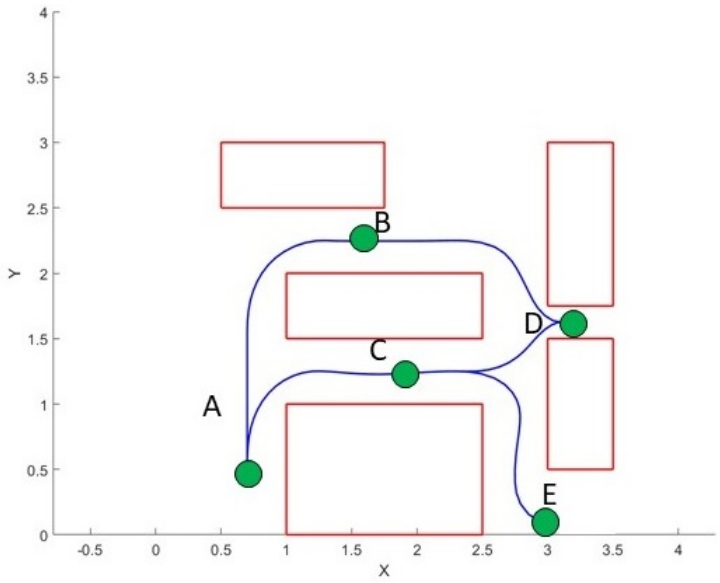


Figure 4.6: The robot plans through a cluttered series of hallways. The terminal condition from the trajectories A-B and A-C are used as initial conditions for the remaining trajectories.

initial conditions. Of note, two statistical outliers have been removed from the histogram of C-E. Both of these runs finished in approximately 20sec. These two runs were most likely hindered by tight constraints around position C. A small variation in the y direction while at position C may result in a nearly infeasible problem by bringing the robot excessively close to obstacles while it is traveling at speed. To confirm this hypothesis, another numerical study is displayed in Figure 4.8, which analyzes outlier occurrence as a function of I.C. variation in the y direction for the path C-E. Here, the bounds of the uniform distribution used to generate the y variation is displayed along the independent axis, and mean computation time and a 2σ bound is displayed along the dependent axis. Notice that the standard deviation is stable up to a 0.25m variation in y. Past this point, both mean and variance increase dramatically and outliers are prevalent. Due to scaling, the last data point is off the dependent axis, but a mean 52.66s and a standard deviation of 121.6s is observed. In addition, two attempts at a solution failed, again strongly correlating the experiments in Fig. 4.7.

The ratio of run times to expected path execution time, including outliers, was 91%. This result confirms that the guaranteed planning algorithm presented here can be solved in a pre-

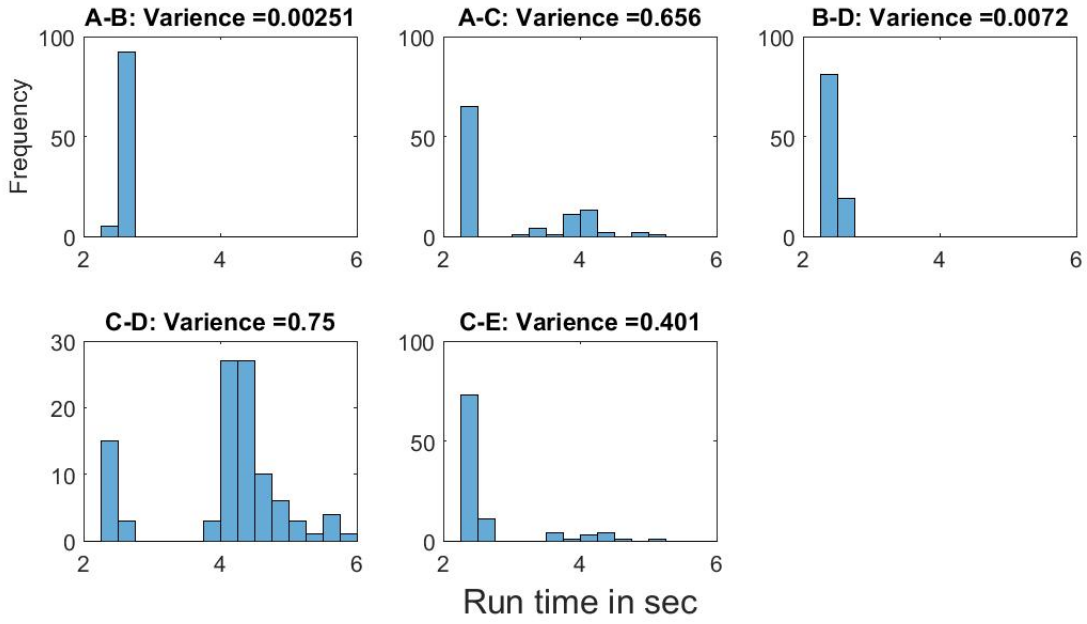


Figure 4.7: A histogram of computation times for each sub-trajectory. Note that two outliers have been removed from C-E for visualization purposes. In addition, A-B has three fewer samples due to failure of the SQP solver to find a feasible path in those cases. The overall success rate is 99.4%.

dictive manner faster than real-time. It is important to note that solve times depend on the initial guess provided to the optimizer, and a sufficiently poor (read infeasible) guess may cause lack of convergence. Three failures were observed in this experiment resulting in a success rate of 99.4%. In all examples the solution of the base-line planner is utilized as an initialization to the guaranteed planner, which works well in practice. Further optimization of the implementation, as well as more direct integration with an SQP solver such as SNOPT, would also help to improve computation times. For example, the current implementation results in additional overhead in SNOPT for each run which add a computational burden of over 0.5s. In addition, utilizing previously optimal trajectories to 'warm start' further computation, along with detailed code optimization, may allow the guaranteed planner to run at several HZ, allowing for dynamic re-planning.

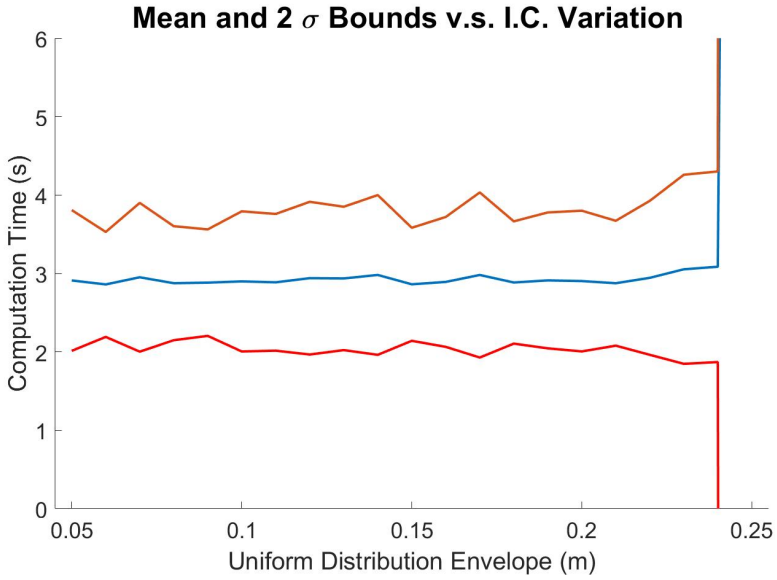


Figure 4.8: The mean computation time of the path from C-E as a function of an increase in initial condition variation. The 2 σ bounds are displayed in red.

4.7.2 Discussion

The results presented above show that the guaranteed planner can be utilized to do near real-time re-planning for low (five) dimensional robotic models. Further optimization and refinement may also significantly reduce computations times. In addition, the techniques presented here can be utilized to generate a library of motion primitives in a sampling based scheme such as [64] which composes motion primitives known to be dynamically feasible. This would allow the application of the algorithm presented here on platforms with limited computation such as rotor craft.

For fixed wing aircraft, this method may be overly conservative. To see this, consider the reactive set $S_r()$ of a fixed wing vehicle about steady level flight. One potential reactive controller could require the robot to go into a loiter pattern towards the direction in which it is currently banking (clockwise or counterclockwise). Clearly, this controller's behavior bifurcates at steady level flight, generating a clockwise or counterclockwise stable path, whose minimal bounding ellipsoids can be drastically different (i.e. do not vary continuously), breaking assumption (10).

Conversely, a less effective controller, which would satisfy the continuity criterion above, could require the robot to always perform a clockwise loiter pattern, which is clearly conservative. Extensions of this work which enable the computationally efficient use of hybrid switching control, for example between different loiter patterns, could eliminate this conservatism.

4.8 Conclusions

The problem of planning under adversarial attack on environmental knowledge is addressed and formulated as an optimal planning and control problem. The motivation behind this formulation was provided by concerns in Cyber Physical Systems security. Thus, this work generates a robust planner capable of accounting for compromised map information.

A novel idea of the *reactive set* is introduced which generalizes stopping distance. This reactive set enables robust planning against unknown static obstacles in the environment while executing minimum time paths. In addition, a visibility constraint of the reactive set is also provided which enables the use of predicted FOVs, generalizing previous work. A connection was made between the continuous time-problem and that of a Mixed Integer Nonlinear Program (MINP), and an asymptotic proof of safety is provided. Tightened constraints are utilized to reduce this MINP to a Nonlinear Program (NLP) enabling an efficient solution using current nonlinear optimization techniques. The reduced formulation was then implemented in simulation.

The behavior of this formulation is analyzed through several examples and statistical timing data is presented. The guaranteed formulation enables a robot to act cautiously before committing to a turning maneuver around a blind corner while also trading between speed and safety. This trade prevents collisions due to uncertainty in the knowledge of the robot's environment. In addition, timing experiments were performed which show the viability of this technique for

real-time optimal planning. Although this work focuses on uncertainty stemming from a malicious attack, the techniques generalize to any cases where environmental knowledge may be incomplete such as areas that have not yet fully been explored.

4.A The MINP Discretization

In this appendix, the connection between the continuous time problem (5) and the NLP formulation is presented; standard discretizations of the objective and dynamics are assumed, and the focus here is on the set inclusion constraints in (5).

The constraint $\exists \tau \ < \ t \quad s.t. \quad S_r(x(t)) \ \subset \ S_s(x(\tau))$, can be read as: 'The reactive set must be observed at some previous time in its entirety.' Suppose a direct optimization method is employed which discretized time into $K \in \mathbb{N}^+$ intervals. The discretized visibility constraint is then:

$$\forall i \in \{2, .., K\} \quad \exists j < i \quad s.t. \quad S_r(x_i) \subset S_s(x_j) \tag{11}$$

This is an integer constraint since it requires an assignment of a previous time step j for each time-step i. Therefore, the presented discretized problem is a Mixed Integer Nonlinear Program (MINP).

$$
\begin{aligned}
\underset{u_i \in U, t_f}{\text{minimize}} \quad & L(x_1, .., x_K, u_1, .., u_K, t_f) \\
s.t. \quad & \zeta_i = 0 \\
& \forall i \in \{1, .., K\} \quad \exists j < i \\
& s.t. \quad S_r(x_i) \subset S_s(x_j) \\
& S_r(x_i) \subset \hat{\mathcal{C}}_{\text{free}} \\
& x_1 = s, \quad x_K = e
\end{aligned}
\tag{12}
$$

4.B Proof of Safety

Given the discretization in (12), it is desirable to show convergence properties for the integer constraint. There have been several works, such as [74], which show that the maximum error in the dynamics violation approaches zero as the number of discretization points $K \to \infty$ due to the defect constraints ζ_i. In the case of (11), it is shown that a fine enough discretization of a path which ensures that (11) holds, also ensures the continuous constraint is satisfied. Several smoothness assumptions are required. Let $\mathcal{E}$ be the set of bounded ellipsoids in $\mathbb{R}^n$.

Assumption 4.B.1 *The reactive and sensor sets are defined respectively by Lipschitz continuous functions:* $S_r : \mathbb{R}^n \to \mathcal{E}, S_s : \mathbb{R}^n \to \mathcal{M}$ *with Lipschitz constants* L_r *and* L_s.

Assumption 4.B.2 *A feasible solution* $x^*(t)$ *to* (5)*, satisfies the set containment constraint loosely in the sense that:* $\exists \epsilon > 0$ *fixed* $s.t. \quad S_r(x(t), t) + B_\epsilon(0) \subset S_s(\tau)$

Given these assumptions:

Proposition 4.B.3 *For any path,* $x^*(t)$*, which satisfies assumptions 4.B.1, 4.B.2, there exists an integer* $M \in \mathbb{N}^+$ *such that* $\forall K \geq M$*, the the discrete set* $\{x_i | x^*(it_f/K) = x_i, i = 0, 1, ..., K\}$ *has the property*

that $S_r(x^*(t)) \subset S_s(x_i)$ *for some* $i \leq Kt/t_f$

Proof:During the following argument, the distance $d(\bullet, \bullet)$ is Euclidean when referring to points and Hausdorff between sets in $\mathbb{R}^n$. Consider an arbitrary point on the path $x^*(t_0)$. Since this point is part of the solution to (5) and satisfies assumption (4.B.2), $\exists 0 \leq t_1 < t_0$ such that $S_r(x(t_0)) + B_\epsilon(0) \subset S_s(x(t_1))$. By assumption (4.B.1), there exists a $\delta > 0$ small enough so that $d(x_1, x_2) < \delta \implies d(S_r(x_1), S_r(x_2)) < \frac{\epsilon}{2}$, and also $d(S_s(x_1), S_s(x_2)) < \frac{\epsilon}{2}$. This comes from taking the maximum of the two functions' Lipschitz constants.

Since $x^*(t)$ is continuous on the interval $t \in [0, t_f]$ it is uniformly continuous and $\Delta t > 0$ can be found s.t.

$$\forall \tau \in [-\Delta t, \Delta t], \quad d(x^*(t), x^*(t \pm \tau)) \leq \delta$$

.

In particular, this is true for $t = t_1$. Thus it follows that $S_r(x^*(t_0) \subset S_s(x^*(t_1 \pm \tau))$. It is clear that $M = \lceil t_f/\Delta t \rceil$ then satisfies the statement of the proposition. ■

Prop. (4.B.3) states that only a finite number of points along a trajectory can guarantee the set inclusion constraint is satisfied along an entire path. Since ϵ can be arbitrarily small, the satisfaction of the continuous constraint is well approximated by the discretization in (11). Note that, if $x(t)$ and $Q(x)$ are Lipchitz, the models provided in section 4.6 ensure that assumptions 4.B.1 and 4.B.2 hold. In addition, Prop. (4.B.3) trivially leads to the following corollary:

Corollary 4.B.4 *Any time discretization of an optimal path* $\{x^*(t_i) | 0 < t_1 < t_2, ..., t_K \leq t_f\}$ *such that* $|t_{i+1} - t_i| \leq \Delta t$*, has the property that* $S_r(x^*(t)) \subset S_s(x^*(t_i))$ *for some* i *where* $t_i < t$

4.C Proof of Lemma (4.6.1)

Proof: The proof follows by an observation and the convexity of the unit circle. By definition, the rays defined by the coordinates $\{x_j^{'}, p_u\}$, $\{x_j^{'}, p_d\}$ are tangent to the circle and can be assumed, *w.o.l.g.*, symmetric about the first axis. Let the translated affine cone $\mathcal{K}$ be that generated by the rays $\overrightarrow{x_j^{'}p_d}$, and $\overrightarrow{x_j^{'}p_u}$, and the point $x_j^{'}$.

$\implies$ By contradiction, suppose an obstacle O is such that $\mathcal{F}(O)$ intersects triangle $\Delta {x'}_j p_u p_d$ but $S_r(x_i)$ is not occluded. Then $\exists y_{\text{obst}} \in \mathcal{F}(O), y_{\text{obst}} \in \Delta {x'}_j p_u p_d$. The triangle is convex and is the formed by the bisection of $\mathcal{K}$. In addition, the arc $\overset{\frown}{p_d, p_u}$ partitions the triangle. Therefore, there exists at least one point z_{arc} on the arc such that $z_{\text{arc}} = \alpha(y_{\text{obst}} - x_j^{'}) + x_j^{'}$. This is a contradiction since, z_{arc} is occluded by the obstacle O and $\mathcal{F}^{-1}(z_{\text{arc}}) \in S_r(x_j)$ is also occluded since affine transformations map lines to lines.

$\impliedby$ Clearly the the planes defined by $x = x_j^{'} + \alpha(p_d - x_j^{'})$, and $x = x_j^{'} + \alpha(p_u - x_j^{'})$, $\alpha \in \mathbb{R}$, are supporting hyperplanes of the convex unit circle. Therefore $B_0(1) \subset \mathcal{K}$. Suppose that no obstacle intersects the triangle $\Delta {x'}_j p_u p_d$. Since constraints 9 are satisfied, no obstacle $\mathcal{F}(O)$ intersects the unit circle. Let $x \in \mathbf{B}_0(1)$ and consider the line segment $s = \{y | y = \alpha x + (1-\alpha) x_j^{'}, \quad a \in (0,1)\}$. The chord (p_u, p_d) bisects the circle and $\Delta {x'}_j p_u p_d$ is convex. Consider the left half-space defined by the affine extension of this chord. Then $\forall x$ in the left half space, inclusive, $s \subset \Delta {x'}_j p_u p_d$. Suppose now $x \in s$ is in the right half-space. Since x is in the cone $\mathcal{K}$, it is parametrized by an angle ψ and a magnitude. Because the chord (p_u, p_d) bisects both the circle and the cone $\exists z \in (p_u, p_d) \quad s.t. \quad z \in s$. By convexity of the circle the portion of s in the right half-space must remain in the circle.

CHAPTER 5

CONCLUSION

The vision of robust cooperative multi-robot autonomy in unknown environments is laudable as a goal of research in robotic autonomy. This vision can be seen as the creation of a robotic village with a division of labor which addresses its sub-problems: exploration, coordination, and objective maximization. As a whole, it is a challenging problem not only because ot requires robust solutions to the problems of information based path planning, control, and estimation, which can be considered entire sub-fields of study, but because the interactions between these separate components are still not well understood and how they should be integrated remains unclear. Consider for example kinematic path planning, which is a mature field with real time capability. Incorporating dynamics and guarantees on localization to accuracy to the planning problem remains a challenge and suffers from scalability. Similarly, localization, estimation, and the underlying information theory are mature fields, but their integration with exploration and planning is still in its relative infancy. Both of these problems are compounded in complexity when multi-agent systems are considered. In other words, the proverbial robotic villagers can move through their environment effectively when they know exactly what dangers it contains. They can also gather information and map an area if a human tells then how to act and where to move.

In the recent two decades, a tremendous amount of progress has been seen in certain sub-problems, especially when concerning the joint problem of Simultaneous Localization and Mapping (SLAM) [citation survey slam] as well as real-time execution of optimal control [49]. In addition these two sub-problems have been deployed on a variety of platforms that are nearing wide commercial deployment [cite slam rumba, autonomous cars, Boston dynamics]. Conversely, the joint problem of information gathering and planning has not seen the same level of progress. This is in part due to the consequences of failure in unknown environments (i.e. loss of the robotic platform or other physical risk). This significant hurtle has been avoided by utilizing

robots with benign dynamics or utilizing conservative control of more capable platforms [75, 64]. This work seeks to contribute to the vision above by addressing three different aspects of these problems which have thus far been elusive. The first problem focuses on guaranteed exploration and localization for ground robots in a partially known environment, and attempts to gain insight into the intersection of localization and planning. The second contribution focuses on the coordination of multiple robotic agents in an exploration and tracking task, and fuses ideas from planning and information gathering. Finally, this work considers guaranteed collision-free minimum time planning for a robotic agent in a partially known environment, and seeks to address the issue of conservatism in motion when the environment is not fully explored.

BIBLIOGRAPHY

[1] A. Davies, "The WIRED Guide to Self-driving cars." https://www.wired.com/story/guide-self-driving-cars/, 2018.

[2] B. Berman, "Whoever Owns the Maps Owns the Future of Self-Driving Cars." http://www.popularmechanics.com/cars/a21609/here-maps-future-of-self-driving-cars, 2016.

[3] A. M. Shkel and V. J. Lumelsky, "The jogger's problem: accounting for body dynamics in real-time motion planning," *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, pp. 441–447 vol.2, 1995.

[4] A. A. Makarenko, S. B. Williams, F. Bourgault, and H. F. Durrant-Whyte, "An experiment in integrated exploration," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 1, pp. 534–539, IEEE, 2002.

[5] L. Freda, F. Loiudice, and G. Oriolo, "A randomized method for integrated exploration," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 2457–2464, IEEE, 2006.

[6] C. Stachniss and W. Burgard, "Exploring unknown environments with mobile robots using coverage maps," in *IJCAI*, pp. 1127–1134, 2003.

[7] W. Burgard, C. Stachniss, and G. Grisetti, "Information gain-based exploration using rao-blackwellized particle filters," in *Proc. of the Learning Workshop*, 2005.

[8] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. A. Castellanos, "Active policy learning for robot planning and exploration under uncertainty.," in *Robotics: Science and Systems*, pp. 321–328, 2007.

[9] R. Sim and N. Roy, "Global a-optimal robot exploration in slam," in *ICRA*, pp. 661–666, IEEE, 2005.

[10] R. Sim, G. Dudek, and N. Roy, "Online control policy optimization for minimizing map uncertainty during exploration," in *ICRA*, vol. 2, pp. 1758–1763, IEEE, 2004.

[11] B. Tovar, L. Munoz-Gómez, R. Murrieta-Cid, M. Alencastre-Miranda, R. Monroy, and S. Hutchinson, "Planning exploration strategies for simultaneous localization and mapping," *Robotics and Autonomous Systems*, vol. 54, no. 4, pp. 314–331, 2006.

[12] M. Juliá, O. Reinoso, A. Gil, M. Ballesta, and L. Payá, "A hybrid solution to the multi-robot integrated exploration problem," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 4, pp. 473–486, 2010.

[13] B. Luders, M. Kothari, and J. P. How, "Chance constrained rrt for probabilistic robustness to environmental uncertainty," in *GNC*, AIAA, 2010.

[14] D. Levine, B. Luders, and J. P. How, "Information-rich path planning with general constraints using rapidly-exploring random trees," in *AIAA Infotech Aerospace Conference, Atlanta, GA*, 2010.

[15] L. Carlone, J. Du, M. K. Ng, B. Bona, and M. Indri, "An application of kullback-leibler divergence to active slam and exploration with particle filters," in *IROS*, pp. 287–293, IEEE, 2010.

[16] L. Carlone and D. Lyons, "Uncertainty-constrained robot exploration: a mixed-integer linear programming approach," in *ICRA*, pp. 1140–1147, IEEE, 2014.

[17] A. Ivanov and M. Campbell, "An efficient robotic exploration planner with probabilistic guarantees," in *ICRA*, pp. 4215–4221, IEEE, 2016.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 1st ed., 2006.

[19] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[20] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *IJRR*, 2009.

[21] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *ICRA*, pp. 5041–5047, IEEE, 2013.

[22] D. Karger, R. Motwani, and G. Ramkumar, "On approximating the longest path in a graph," *Algorithmica*, vol. 18, no. 1, pp. 82–98, 1997.

[23] C. Berge and A. Ghouila-Houri, *Programming, Games and Transportation Networks*. Methuen Publishing, 1965.

[24] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[25] D. P. Bertsekas, *Dynamic Programming and Optimal Control. 3rd ed.* Athena Scientific, 2007.

[26] J. Pearl, *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley, 1984.

[27] G. Zhang, S. Ferrari, and M. Qian, "An information roadmap method for robotic sensor path planning," *Journal of Intelligent and Robotic Systems*, vol. 56, no. 1-2, pp. 69–98, 2009.

[28] J. Pineau, G. Gordon, S. Thrun, *et al.*, "Point-based value iteration: An anytime algorithm for pomdps," in *IJCAI*, vol. 3, pp. 1025–1032, 2003.

[29] G. Klančar and I. Škrjanc, "Tracking-error model-based predictive control for mobile robots in real time," *Robotics and Autonomous Systems*, vol. 55, no. 6, pp. 460–469, 2007.

[30] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 743–761, April 2012.

[31] A.-a. Agha-mohammadi, S. Chakravorty, and N. M. Amato, "Firm: Feedback controller-based information-state roadmap-a framework for motion planning under uncertainty," in *IROS*, pp. 4284–4291, IEEE, 2011.

[32] M. F. Huber, T. Bailey, H. Durrant-Whyte, and U. D. Hanebeck, "On entropy approximation for gaussian mixture random vectors," in *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pp. 181–188, Aug 2008.

[33] A. Knutson and T. Tao, "Honeycombs and sums of hermitian matrices," *Notices Amer. Math. Soc*, vol. 48, no. 2, 2001.

[34] C. Robin and S. Lacroix, "Multi-robot target detection and tracking: taxonomy and survey," *Autonomous Robots*, vol. 40, no. 4, pp. 729–760, 2016.

[35] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics A survey," *Autonomous Robots*, vol. 31, no. 4, pp. 299–316, 2011.

[36] A. Ivanov and M. Campbell, "An efficient robotic exploration planner with probabilistic guarantees," in *Proceedings - ICRA*, vol. 2016-June, (Stockholm, Sweden), pp. 4215–4221, 2016.

[37] M. Al khawaldah and A. Nuechter, "Multi-Robot Cooperation for Efficient Exploration," *Automatika*, vol. 55, no. 3, pp. 276–286, 2014.

[38] R. Mottaghi and R. Vaughan, "An integrated particle filter and potential field method applied to cooperative multi-robot target tracking," *Autonomous Robots*, vol. 23, pp. 19–35, jul 2007.

[39] L. C. A. Pimenta, M. Schwager, Q. Lindsey, V. Kumar, D. Rus, R. C. Mesquita, and G. A. S.

Pereira, "Simultaneous Coverage and Tracking (SCAT) of moving targets with robot networks," *Springer Tracts in Advanced Robotics*, vol. 57, pp. 85–99, 2010.

[40] J. Elston and E. W. Frew, "Hierarchical distributed control for search and tracking by heterogeneous aerial robot networks," *Proceedings - ICRA*, pp. 170–175, 2008.

[41] H. Wei and S. Ferrari, "A geometric transversals approach to sensor motion planning for tracking maneuvering targets," *IEEE TAC*, vol. 60, no. 10, pp. 2773–2778, 2015.

[42] D. Levine, B. Luders, and J. How, "Information-Theoretic Motion Planning for Constrained Sensor Networks," *JAIS*, vol. 10, no. 10, pp. 476—-496, 2013.

[43] T. K. Yaakov Bar-Shalom, Xiao-Rong Li, *Estimation with applications to tracking and navigation*. New York: Wiley, 1st ed., 2001.

[44] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, Massachussetts: The MIT Press, 2005.

[45] T. Zhou, "Asymptotic Behavior of Recursive State Estimations with Intermittent Measurements," *IEEE Transactions on Automatic Control*, vol. 61, no. 2, pp. 400–415, 2016.

[46] A. Censi, "Kalman filtering with intermittent observations: Convergence for semi-Markov chains and an intrinsic performance measure," *IEEE Transactions on Automatic Control*, vol. 56, no. 2, pp. 376–381, 2011.

[47] K. Plarre and F. Bullo, "On Kalman filtering for detectable systems with intermittent observations," *IEEE Transactions on Automatic Control*, vol. 54, no. 2, pp. 386–390, 2009.

[48] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, Massachusetts: Athena Scientific, 3 ed., 2005.

[49] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Philadelphia: SIAM, 2nd ed., 2009.

[50] A. M. Bayen, I. M. Mitchell, M. K. Osihi, and C. J. Tomlin, "Aircraft Autolander Safety Analysis Through Optimal Control-Based Reach Set Computation," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 1, pp. 68–77, 2007.

[51] M. E. Campbell and N. R. Ahmed, "Distributed Data Fusion: Neighbors, Rumors, and the Art of Collective Knowledge," *IEEE Control Systems*, vol. 36, no. 4, pp. 83–109, 2016.

[52] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[53] Tedrake, Russ. and the Drake Development Team, "Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems," 2016.

[54] K. Zetter, "Iran: Computer Malware Sabotaged Uranium Centrifuges." https://www.wired.com/2010/11/stuxnet-sabotage-centrifuges/, 2010.

[55] S. Checkoway, D. Mccoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," *System*, pp. 6–6, 2011.

[56] D. Majumdar, "Iran Claims Successful Test Flight of Stealth UAV." https://news.usni.org/2014/11/12/iran-claims-successful-test-flight-stealth-uav, 2014.

[57] M. Yilin, C. Rohan, and S. Bruno, "Detecting Integrity Attacks on SCADA Systems," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 11239–11244, 2011.

[58] F. Pasqualetti, F. Dorfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 11, pp. 2715–2729, 2013.

[59] H. Fawzi, P. Tabuada, and S. Diggavi, "Secure estimation and control for cyber-physical systems under adversarial attacks," *IEEE Transactions on Automatic Control*, vol. 59, no. 6, pp. 1454–1467, 2014.

[60] M. Pajic, I. Lee, J. G. Pappas, and G. J. Pappas, "Attack-Resilient State Estimation for Noisy Dynamical Systems," *IEEE Transactions on Control of Network Systems*, vol. PP, no. 99, pp. 1–10, 2016.

[61] Y. Shoukry, A. Puggelli, P. Nuzzo, A. L. Sangiovanni-vincentelli, S. A. Seshia, and P. Tabuada, "Sound and Complete State Estimation for Linear Dynamical Systems Under Sensor Attacks Using Satisfiability Modulo Theory Solving," *2015 American Control Conference (ACC)*, pp. 3818–3823, 2015.

[62] N. Dadkhah and B. Mettler, "Survey of motion planning literature in the presence of uncertainty: Considerations for UAV guidance," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 65, no. 1-4, pp. 233–246, 2012.

[63] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey," *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.

[64] B. T. Lopez and J. P. How, "Aggressive collision avoidance with limited field-of-view sensing," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-Septe, pp. 1358–1365, 2017.

[65] P. E. Missiuro and N. Roy, "Adapting probabilistic roadmaps to handle uncertain maps," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, no. May, pp. 1261–1267, 2006.

[66] M. P. Vitus and C. J. Tomlin, "A hybrid method for chance constrained control in uncertain environments," *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 2177–2182, 2012.

[67] A. Girard, "Reachability of uncertain linear systems using zonotopes," *Hybrid systems: computation and control*, pp. 291–305, 2005.

[68] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 57, no. 1-4, pp. 65–100, 2010.

[69] P. Sujit, S. Saripalli, and J. B. Sousa, "Unmanned Aerial Vehicle Path Following: A Survey and Analysis of Algorithms for Fixed-Wing Unmanned Aerial Vehicless," *IEEE Control Systems*, vol. 34, no. 1, pp. 42–59, 2014.

[70] E. Scholte and M. E. Campbell, "Robust nonlinear model predictive control with partial state information," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 4, pp. 636–651, 2008.

[71] E. Asarin, T. Dang, and A. Girard, "Hybridization methods for the analysis of nonlinear systems," *Acta Informatica*, vol. 43, no. 7, pp. 451–476, 2007.

[72] A. Ivanov, "Robotic planning resilient to map attack." `https://github.com/A-I-Ivanov/CPS_Untrusted_Maps`, 2018.

[73] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[74] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of Guidance Control and Dynamics*, vol. 10, no. 4, pp. 338–342, 1987.

[75] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient bipedal robots based on passive-dynamic walkers," *Science*, vol. 307, no. 5712, pp. 1082–1085, 2005.